A Framework for the Systematic Evaluation of Malware Forensic Tools

Thesis

A framework for the systematic evaluation of malware forensic tools

Ian Kennedy BEng(Hons), PGCE, MBCS, CITP, CEng

A thesis submitted to The Open University
for the degree of
Doctor of Philosophy in Computing

School of Computing and Communications
Faculty of Science, Technology, Engineering and Mathematics
The Open University
Milton Keynes, UK

March 2017

Declaration

Some of the material in this thesis has previously been published in the following areas:

Book chapters

Bryant, R. and Kennedy I. (2014) 'Investigating Digital Crime', in Bryant, R. and Bryant, S. (eds.) *Policing Digital Crime*, Farnham: Ashgate, pp. 123-145

Kennedy I. and Day, E. (2014) 'Procedures at Digital Crime Scenes', in Bryant, R. and Bryant, S. (eds.) *Policing Digital Crime*, Farnham: Ashgate, pp. 147-160

Kennedy I. and Day, E. (2014) 'Digital Forensic Analysis', in Bryant, R. and Bryant, S. (eds.) *Policing Digital Crime*, Farnham: Ashgate, pp. 161-185

Bryant, R., Day, E. and Kennedy I. (2014) 'Opportunities and Challenges for the Future', in Bryant, R. and Bryant, S. (eds.) *Policing Digital Crime*, Farnham: Ashgate, pp. 201-217

Kennedy, I. (2008) 'Investigating Digital Crime', in Bryant, R. and Bryant, S. (eds.) *Investigating Digital Crime*. Chichester: John Wiley & Sons Ltd., pp. 49-78

Podcast

Kennedy, I. (2011) *The Trojan Defence.* [Podcast]. 18 May. Available at: https://itunes.apple.com/gb/itunes-u/the-trojan-defence-audio/id438692522?mt=10 (Accessed: 20 February 2017).

Conference paper

Kennedy, I. (2010) 'Towards scientific malware analysis' *Proc. 4th International Conference on Cybercrime Forensics Education & Training*, Canterbury Christ Church University, Canterbury, 2-3 September 2010

Magazine article

Kennedy, I. (2010) 'Playing with fire: Dissecting malicious software' *Digital Forensics Magazine* (May), pp. 45-51. Available at: http://digitalforensicsmagazine.com/index.php?option=com_content&view=article&id=415:malware&catid=40:issue3&Itemid=89 (Accessed: 20 February 2017).

All of the work presented in this thesis describes my original contributions, except where otherwise explicitly stated and referenced.

Acknowledgements

I would like to offer my sincere thanks to my supervisors Dr Arosha Bandara and Blaine Price. This research has been a long and difficult journey with many challenges from different quarters. Without their steadying support this work would not have made it to completion. I would also like to extend my gratitude to Dr Thein Tun for his comments and feedback on this dissertation. I am also indebted to Professor Marian Petre, whose insightful input is inspiring in its surgical precision!

Abstract

Following a series of high profile miscarriages of justice linked to questionable expert evidence, the post of the Forensic Science Regulator was created in 2008 with a remit to improve the standard of practitioner competences and forensic procedures. It has since moved to incorporate a greater level of scientific practice in these areas, as used in the production of expert evidence submitted to the UK Criminal Justice System. Accreditation to their codes of practice and conduct will become mandatory for all forensic practitioners by October 2017. A variety of challenges with expert evidence are explored and linked to a lack of a scientific methodology underpinning the processes followed. In particular, the research focuses upon investigations where malicious software ('malware') has been identified.

A framework, called the 'Malware Analysis Tool Evaluation Framework' (MATEF), has been developed to address this lack of methodology to evaluate software tools used during investigations involving malware. A prototype implementation of the framework was used to evaluate two tools against a population of over 350,000 samples of malware. Analysis of the findings indicated that the choice of tool could impact on the number of artefacts observed in malware forensic investigations as well as identifying the optimal execution time for a given tool when observing malware artefacts.

Three different measures were used to evaluate the framework. The first of these evaluated the framework against the requirements and determined that these were largely met. Where the requirements were not met these are attributed to matters either outside scope or the fledgling nature of the research. Another measure used to evaluate the framework was to consider its performance in terms of speed and resource utilisation. This identified scope for improvement in terms of the time to complete a test and the need for more economical use of disk space. Finally, the framework provides a scientific means to evaluate malware analysis tools, hence addressing the Research Question subject to the level at which ground truth is established.

A number of contributions are produced as the output of this work. First there is confirmation for the case for a lack of trusted practice in the field of malware forensics. Second, the MATEF itself, as it facilitates the production of empirical evidence of a tool's ability to detect malware artefacts. A third contribution is a set of requirements for establishing trusted practice in the use of malware artefact detection tools. Finally, empirical evidence that supports both the notion that the choice of tool can impact on the number of artefacts observed in malware forensic investigations as well as identifying the optimal execution time for a given tool when observing malware artefacts.

TABLE OF CONTENTS

List of Tables

List of Figures

Main Body of thesis

List of Cases cited

Parties	Year	Citation	Description
R v Shepard	1992	[1993] AC 380	Expertise required to present digital evidence from shop till roll
Daubert v Merrell	1993	Daubert v Merrell 5 Dow Pharmaceuticals Inc. 509 U.S. 579 (1993)	Defined the standard for admitting expert testimony in US federal courts, which came to be known as the Daubert Standard
R v Clark	1995	R v Clarke (RL) [1995] 2 Cr. App. R. 425	Determined that it would be wrong to deny justice access to new and fledging techniques
R v Clark	1999	[2000] EWCA Crim 54	Challenge to expert evidence infanticide case
R v Cannings	2002	[2004] EWCA Crim 1	Challenge to expert evidence infanticide case
R v Caffrey	2003	Unreported (The Times, 2003)	Trojan defense applied in a DoS attack case
R v Green	2003	[2004] EWCA Crim 2795	Challenge to expert evidence infanticide case
R v Patel	2003	Unreported (Studd, 2003)	Challenge to expert evidence infanticide case
R v Schofield	2003	Unreported (Gibb, 2003)	Trojan defense applied in an indecent images of children case
Williford v State of Texas	2004	Williford v State of Texas (2004)	Forensic software EnCase deemed to be reliable, based on anecdotal testimony of one police officer
Sanders v State of Texas	2006	Sanders v State of Texas, (2006)	Court set precedent concerning scientific reliability of a specific methodology
State of Florida v. Casey Marie Anthony	2011	State of Florida v Casey Marie Anthony	Discrepancies identified between two Internet history tools used to produce expert testimony
Lundy v R	2013	[2013] UKPC 28	Use of novel evidence

Table 0-1 : List of cases cited

Glossary of Terms

Term	Description
Admissible	Refers to an exhibit or testimony that is permitted as evidence in a court.
Analysis script	The component of the MATEF used to analyse Tool Log Files.
Application Program Interface	A set of functions and procedures that allow software developers to write code to access the features or data of an operating system, application, or other service, such as connecting to the Internet.
Artefact	A change in data that arises as a result of an action performed on a computer, such as the creation of a file.
Assembly language	Low level instructions that perform limited stepwise operations on device's memory locations.
Asymptotic significance	A computationally less intensive method of calculating the significance value, valid for large samples.
Bare metal platform	An environment where the operating system is installed on physical machine hardware (like a regular desktop computer).
Bi-modality	A distribution containing two peaks.
Binary file	A computer file that is not a text file. Binary files typically contain either data or executable code.
Binomial sign test	A nonparametric test similar to the *Wilxocon signed rank test*.
Black box testing	A behavioural software testing method in which the internal structure and algorithm of the item being tested is not known to the tester.
Command-line interface	A terminal based means to send commands to and receive output from a software program.
Cybercrime	Criminal activity that is focused upon or makes use of the Internet or a digital device.
Cybersecurity	The field of security concerned with protecting computers, networks, programs and data from unauthorized access or attacks.
Cyberspace	The realm within which electronic communications occur.
Debug	A process of located faults in software.
Demilitarised zone	A network located outside an organisation's main firewall typically containing servers accessible to the Internet.
Dependent t-Test	A statistical test that compares the means of two related groups of data.
Digital forensics	The analysis of artefacts located on electronic devices and networks as part of an investigation, typically for court.
Disassembly	The process of examining a binary executable file to produce assembly language instructions.
Distributed Denial of Service attack	A form of attack whereby unauthorised access to multiple devices is gained with a view to using these to coordinate multiple requests to a target device and thereby render it inaccessible to others.

Dual-tool verification	The process of comparing the output of one tool with another.
Dynamic Link Library	A binary file containing commonly used code used by applications, such as code to display an open file dialogue window.
Dynamic malware analysis	The process of analysing malware behaviour by executing the binary file in a controlled environment.
Effect size	A measure of how easy it is to observe a given effect.
Emulation platform	Similar to a *Virtualisation platform*, this is an environment where all of the hardware is emulated in software, allowing for the *Host* and *Guest* architectures to be different.
Expected value	The value expected as the result of the counting artefacts of a given type, e.g.: count of open ports
Expert evidence	Evidence that requires an expert to produce and interpret the evidence in a manner that meets a court's admissibility requirements.
Familywise error	A type of error introduced as a result of combining the results of multiple independent statistical tests.
Forensic Science Regulator	The office created in 2008 by the U.K. Government to oversee all forensic science provision within the U.K. criminal justice system.
Hash	The value returned from a hash function, which calculates a value of fixed size from any size of data.
Internet simulator	The component of the MATEF used to simulate Internet services such as email and web servers.
Interval data	Data with the same properties as *Ordinal data*, but the size of any difference can be determined.
Kolmogorov-Smirnoff test	An implementation of a *Normality test*.
Malware	Short for malicious software and typically disrupts, gathers sensitive information from or gains unauthorised access to computer systems.
Malware artefact database	The database component of the MATEF used to store malware artefacts.
Malware forensics	The analysis of malware as part of a digital forensic investigation.
Malware library	A folder structure housing the entire malware binary population available to the MATEF during testing.
Malware source	The source component of the MATEF used to acquire malware binaries.
McNemar test	A nonparametric test similar to the *Wilxocon signed rank test* that only supports nominal data.
Message Digest 5	A form of *hash* where a specific algorithm is used to verify data integrity.
Normality test	A form of statistical test used to determine if a dataset can be modelled by a Normal distribution.
Nominal data	Data that can only be categorised or assigned a label/name.
Nonparametric tests	The converse of a *parametric test* whereby no assumptions are made that sample data comes from a population that follows a probability distribution based on a fixed set of parameters.

Observed value	The value obtained as the result of the count of artefacts of a given type, e.g.: count of open ports
Oracle	The component of the MATEF used to estimate the ground truth value of expected artefacts.
Ordinal data	Data with the same properties as *Nominal data*, but can also be ordered or compared, based on ranking or size.
Packer	Software that compresses an original binary file (typically malware) to render the original code and data unreadable.
Parametric test	A type of statistical test that assumes that sample data comes from a population that follows a probability distribution based on a fixed set of parameters.
Port	An interface of a computer. Each port has a number that is typically associated with a specific means of communication, e.g.: Internet browsers typically use port 80.
Portable Executable file format	A standard binary file format containing executable code that is recognised by the Windows operating system.
Prefetch	Files created to speed up the launching of applications on a Windows computer.
Process	An instance of a computer program that is being executed.
Ratio data	Data with the same properties as *Interval data*, but the factor/ratio of any difference can be determined.
Registry	The Registry is a hierarchical database that stores low-level settings for the Microsoft Windows operating system
Registry key	Part of the structure used to store data in the Windows Registry. Similar to folders on a disk.
Repeatable	The closeness of agreement between independent test results obtained under the same conditions by the same operator within a short interval of time.
Reproducible	The closeness of agreement between independent test results obtained under the same conditions by the different operators in different locations with different equipment.
Restore point	A backup feature of the Windows operating system that allows the user to revert a computer's state to that of a previous point in time.
Reverse engineering	The process of examining a binary executable file and identifying the commands and algorithm used to determine how it operates.
Sandbox	A virtual space in which new or untested software or coding can be run securely.
Shapiro-Wilk test	An implementation of a *Normality test*.
Standard error	A measure of population dispersion, meaningful only to data distributed symmetrically about a mean.
Standardised test statistic	The *test statistic* expressed in units of standard deviation
Static malware analysis	The process of analysing malware through the examination of the binary file without executing it.

Test control script	The script used by the MATEF to manage the test process. Typical tasks include starting and reverting VMs.
Test environment	The *virtualisation platform* in which malware binaries and tools under test are executed on the MATEF.
Test statistic	The ratio of systematic to unsystematic (or effect to error), if the null hypothesis is true. Arbitrary for unknown distributions.
Tool log files	Log files generated by tools under test.
Trojan defence	A defence offered by a defendant whereby an alleged offence was performed as a result of some form of malware (or third-party) that gained control of their computer.
User Access Control	A security feature of the Windows operating system that prevents unauthorized changes to your computer.
Validation	The process of generating independent evidence that a method, process or device is fit for purpose. Answers the question "Is it the right method, device, etc.?"
Verification	Confirmation through comparing with an independent source that a method, process or device is fit for purpose. Answers the question "Are we doing it right?"
Virtual machine	An environment where the operating system (called a Guest) is installed in an environment controlled by software (called a Hypervisor) running on an operating system on a physical machine (called a Host).
Virtualisation platform	An environment that uses a *virtual machine*.
Wilcoxon signed rank test	A nonparametric test equivalent to the *Dependent t-Test*.
Wrapper	A short item of plug-in code containing tool specific parameters.

List of Abbreviations

Acronym	Meaning
API	Application Program Interface
CFReDS	Computer Forensic Reference Data Sets
CFTT	Computer Forensics Tool Testing
CJS	Criminal Justice System
CLI	Command-Line Interface
COTS	Commercial Off The Shelf
CPS	Crown Prosecution Service
CPU	Central Processing Unit
CSV	Comma Separated Value
DC3	Department of Defence Cyber Crime Centre
DDoS	Distributed Denial of Service
DFTT	Digital Forensic Tool Testing
DLL	Dynamic Link Library
DMZ	Demilitarised zone
DNS	Domain Name System
FSR	Forensic Science Regulator
HTTP	Hypertext Transfer Protocol
IRC	Internet relay chat
MATEF	Malware Analysis Tool Evaluation Framework
MD5	Message Digest 5
NIST	National Institute of Standards and Technology
OLAF	European Anti-Fraud Office
PE	Portable Executable
PMS	Program Manager Script
RAM	Random Access Memory
SMTP	Simple Mail Transfer Protocol
SWGDE	Scientific Working Group on Digital Evidence
UAC	User Access Control
VM	Virtual Machine
VMM	Virtual Machine Manager

Chapter 1 Introduction

The biological virus has been mankind's constant companion throughout history. Unseen by the naked eye and adaptable to its environment, a virus can be harmful, hostile and very capable of defending itself. The battle to eradicate it is a never-ending arms race between mankind and the virus.

Unlike its biological counterpart, computer viruses are man-made but share many of the characteristics and challenges in their handling and study. As with the biological variety, computer viruses can be hostile in nature and hazardous to handle for analysis purposes. Although often termed *computer virus*, the more definitive term is *malicious software* (a.k.a. *malware*). This is due to fact that the infiltration and distribution techniques they use have evolved beyond those used by biological viruses and that they are typically synthesised with hostile intent.

Despite the numerous and conflicting (British Computer Society, 2011) anti-malware reports routinely published by vendors, the exact number of species is unknown and is subject to an increasing number of variants (Smith, 2014) which makes obtaining an accurate assessment of the threat level at any one time difficult. Smith (2014) argues that estimates of exact numbers of infections are dependent on either statistics reported by security and anti-malware vendors or the monitoring of Internet traffic. The former is subject to issues of sample sizes and bias while the latter has data attribution issues, whereby Internet traffic monitoring logs don't always contain the information needed to attribute network activity to specific malware. Aycock (2006) argues the single biggest problem is that there is no industry-wide agreement on what constitutes a threat. He also points out the figures quoted in reports are only for the known instances of malware and that it is impossible to known how many unknown threats are in the wild.

Despite these ambiguities, Baker *et al.* (2011) identify malware as a cybersecurity issue; they report almost two-thirds of critical infrastructure companies admit to finding malware on a monthly basis designed to sabotage their systems. Hence, although there are ambiguities surrounding the quantification and classification of malware, it is recognised that malware remains a key vector for cybersecurity attacks. This is supported by Hunton (2012) who cites the revelation from the Cabinet Office (2010) that cybersecurity is one of the highest priority national security risks to the UK. Such is the scale of concern of this threat that in 2011 the UK Government published a Cybersecurity strategy (Cabinet Office, 2011). In 2016 The UK Government reaffirmed its position in their updated strategy that the cyber threat continues to be a "Tier One risk to UK interests" (Cabinet Office, 2016).

Underpinning a nation's commerce and military sectors, the strategy goes on to argue that the increasing use of cyberspace *"*means that its disruption can affect nations' ability to function effectively in a crisis*"*.

The impact of this reaches beyond national security matters and into domestic law enforcement capability. Burd *et. al*. (2011) recognise that national security and law enforcement agencies have historically evolved their capability along differing paths. They go on to argue that the increasing sophistication of cybercrime now supports a need to bridge the capability gap between them.

The need for law enforcement to increase their capability followed comments reported in Computer Weekly (Grant, 2010) when the then Metropolitan Police commissioner, Paul Stephenson is reported to have declared that the skills available to his cyber investigators were "thin compared to the skills at the disposal of cyber criminals". The National Audit Office corroborated this viewpoint and asserted it could take 20 years to address the cyber-security skills gap (2013). Hunton (2012) admits that law enforcement is in a position where in terms of the specialist knowledge needed to investigate the evolving cybercrime domain, demand is in excess of capability. Similarly, Runciman (2011) identifies malware related cybercrime as a specific area where law enforcement need to be better resourced.

It is not uncommon during cybercrime investigations to discover malware. The presence of malware on a computer will either be intentional or unintentional on the part of the suspect. In the case of the former, the suspect may have either created the malware or obtained it from a third party, possibly with a view to committing an offence, such as DDoS attack or unauthorised access to a computer system.

For the latter case, the suspect may be an actual or potential victim of crime in that if executed, the malware will likely perform one or more actions, such as granting unauthorised access to their computer, exfiltrating personal data or using their computer to attack or access a remote computer without authorisation. For an individual under investigation, a common tactic is to claim the alleged illegal activity was performed as a result of some form of malware (or third-party) that gained control of their computer (Bridges, 2008). This is referred to as the *Trojan defence*.

Regardless of the intentions of the suspect in possessing malware on their computer, both civil and criminal forensic practitioners have a duty to identify the capabilities of any malware found as part of an investigation. In the UK, the forensic practitioner is reminded of their responsibility of their duty to the court under both the Civil Procedure Rules (Ministry of Justice, 1999) and the Criminal Procedure Rules (Ministry of Justice, 2015).

To undertake this duty, the forensic practitioner is reliant on their tools, skills and knowledge of malware to detect, identify and study the behaviour of any identified malware. As a result, the forensic practitioner aims to form an opinion on the impact any identified malware has on an investigation.

1.1 Justification

The terms *malware* and *forensics* are increasingly being combined to describe the emerging field *malware forensics*. The original motivation for this thesis arose from the realisation that digital forensic practitioners were conducting malware forensic investigations in a largely anecdotal manner.

Court proceedings involving malware that is not properly investigated inevitably become a candidate for miscarriages of justice, as the court would be forming a judgment without being fully informed of the facts.

An investigation involving malware, however, is just one example of where the expert's opinion, findings and associated methodologies are subject to an increasing level of scrutiny due to recent problems with expert evidence. These are examined in more detail in the sections that follow.

1.1.1 The Trojan defence

Separating user actions from those of malicious software is the fundamental objective when investigating the Trojan defence. The impact of this defence is illustrated by the following cases. As a result of a criminal investigation, malicious software, described as a *Trojan horse* was found alongside a number of indecent images of children on the computer belonging to Karl Schofield. A forensic expert at the trial of R v Schofield [2003][1] concluded that it was the Trojan horse and not the actions of the defendant that led to the pictures being downloaded (GetReading, 2003). Similarly, in R v Green [2003] the defendant was acquitted for downloading indecent images of children after it was argued that the material could have been placed there by one of eleven items of malware (described as *Trojan horses*) found on his computer. A few months later in R v Caffrey [2003], the defendant was also acquitted as he successfully argued that it was the actions of a Trojan horse that launched a Distributed Denial of Service (DDoS) attack from his computer on the Port of Houston, Texas.

[1] For all legal case citations, see Table 0 1

Similarly, Amero (2007) and Fiola (2008) both involved a defence citing malware as the cause of all or part of their alleged actions. More recently Welham (2010) reported on the case of Chris Singam who was acquitted of making and possessing indecent images of children as a result of a "virus that meant he could not have known indecent images of children were being sent to his computer." In cases such as these, the Defence will typically argue in terms of possibilities (and hence introduce reasonable doubt) while the Prosecution focus on likelihoods (and how low such likelihoods are in their personal experience). Brown (2015) highlights the Trojan Defence as one of several tactics used by counsel to raise doubt as to the authenticity of the electronic evidence presented to court. On the matter of malware behaviour, neither side present anything other than anecdotal evidence to support their stance.

A more comprehensive review of these and other cases covering 2003 to 2013 is provided by Bowles and Hernandez-Castro (2015) who highlight "clear and obvious mistakes" with regard to Trojan Defence cases over a 10 year period.

From a sceptic's perspective, the Trojan defence is not an issue; conventional artefacts are sufficient to determine if the identified actions were performed by malware, or intentionally by the user (Carvey, 2009). However, anti-forensic measures (commonly adopted by malware) are cited as a risk to this practice (Kessler, 2007), (Casey, 2002). There is also an argument that sceptics will place too much trust in their own anecdotal experience of repeated confirmation that malware was not the cause of illegal activity found on a computer.

1.1.2 Unfounded trust in repeated confirmation

When asked about the possibility that malware has been used to perform certain types of operations (such as downloading child abuse images), some digital forensic experts defer to their own anecdotal experience and assert arguments based on the fact that they have "yet to see an example" of such behaviour by malware (McLinden, 2009). Others have made greater, albeit non-scientific, attempts to reach out to the practitioner community to locate any instances of such malware and reported that they "haven't seen a single case" (Douglas, 2007). Arguments such as these can be convincing in court but are based on inductive reasoning derived from repeated confirmation. Hence it is possible for this statement to be proven incorrect the moment a single instance of malware downloading child abuse images is identified. Whilst inductive reasoning is useful to use a small number of observations to infer a larger theory or generate a hypothesis, it cannot be used to test scientific theory. This means the use of repeated confirmation is not scientific in its approach.

Similarly, the results from mainstream digital forensic tools have been accepted "based solely on the reputation of the vendor" (Garfinkel, Farrell, Roussev & Dinolt, 2009). Repeated confirmation, such as this, does not prove anything. Criminals are reported to have exploited this viewpoint to hide contraband material (McLinden, 2011). It may be argued that anecdotal arguments are sufficient for legal proceedings and accepted by Courts. However, some see this simply as the result of the Court's naivety in the area of forensic science (Saks & Faigman, 2008). Such naivety, it could be argued, has led to the discovery of problems with some expert evidence.

1.1.3 Recent problems with expert evidence

Recent high profile miscarriages of justice have been attributed in part to flawed expert evidence (Law Commission, 2011). The Solicitors Journal (2011) cites the Law Commissioner, Professor David Ormerod, as saying that judges are "in the unsatisfactory position of having no real test to gauge the unreliability of expert evidence".

The case of R v Clark [1999] concerned the circumstances surrounding infant cot deaths. Professor Sir Roy Meadows made a number of claims that had "no statistical basis" (Royal Statistical Society, 2001). As a paediatrician (not a statistician), Meadows was testifying outside of his expertise. Similar claims were made by Meadows in the subsequent trials of R v Cannings [2002] and R v Patel [2003]. All of these convictions were quashed at subsequent appeals and the Law Commission reviewed the admissibility of expert evidence for use in criminal trials (Law Commission, 2011). The report called for a move to incorporate a greater level of scientific principles and provenance in expert evidence.

1.1.4 Lack of scientific principles

One of the challenges in applying greater levels of scientific rigour to expert evidence derived from forensic science is the view that forensic science is an oxymoron, lacking the scientific principles enjoyed by established scientific disciplines (Kennedy, 2003). Some disciplines such as forensic otoscopy (Mohurle, Khutwad, Kunjir & Bhosle, 2016), which seeks to identify humans based upon their ear impression, have little formal research and no research agenda. A view taken when such disciplines are applied is that there is a correlation between "dubious forensic science and wrongful convictions" (Cooley, 2004). Cole (2011) echoes this view and points to a lack of sufficient studies in some disciplines of forensic science, such that little can be inferred about their accuracy.

This absence of a body of knowledge, established through accepted scientific methodologies, has led to criticism of practitioners being rhetorical in their application of substance or methodology (Saks & Faigman, 2008). Saks & Faigman go on to state that scientific

principles, such as rigorous empirical testing, inductive methodologies and reporting of error rates are all absent from many of the "non-science forensic science" disciplines.

Without this scientific pedigree, many of the specialties within forensic science taken into the courtroom face the risk of being labelled as *junk science* (Huber, 1993). Epstein (2009) cites fingerprints, handwriting and firearms as three examples of such science. He goes on to promote the exclusion of such evidence from trials. Other examples include voice identification, footprints, bite marks, tool marks, blood spatter and hair comparison (Edmond, Biber, Kemp & Porter, 2009). Broadly speaking, all of these specialties concern themselves with applying individualization to link an artefact to a suspect.

Computers are meticulous keepers of time and they record times and dates for a multitude of events that take place on them. Specialities such as computer forensics and malware forensics utilise this intrinsic auditing feature to determine the provenance of identified artefacts. It is ironic that these specialties themselves also do not have any such scientific provenance.

The lack of a scientific footing for malware forensics has a greater impact for the discipline than it does for computer forensics. The availability of both undergraduate and post-graduate qualifications in computer forensics provides an opportunity for practitioners to engage with their discipline on an academic and scientific footing. Although included as modules on some courses, there are no such equivalent academic qualifications for malware forensics.

This absence of both a scientific and academic foundation identifies a number of risks for evidence tendered in criminal proceedings. Malware is designed to obfuscate its true intentions and hinder attempts to analyse it (Wagener, Dulaunoy & Engel, 2008). There is therefore a level of uncertainty associated with any conclusions drawn from malware analysis. This uncertainty can be used to raise reasonable doubt about the true nature and intentions of malware.

There is also uncertainty in the ontology of the field. Aycock (2006) argues there is no universally accepted definition of terms such as *virus*. This is echoed by Bureau and Harley (2008), who suggest the expectations of end users are too high. They go on to suggest it is too impractical to classify malware by names. Mundie and McIntire (2013) also identify issues with inconsistent vocabulary amongst anti-malware vendors and members of the cybersecurity community.

The complexity of the subject matter and the specialist skills required to study it (e.g.: reverse engineering & assembly language) may make the specialty less accessible to practitioners.

Lawyers seeking to undermine evidence produced from malware analysis currently have a rich choice of attack vectors they can use to introduce reasonable doubt concerning its validity. The lack of scientific provenance, the skillset of the practitioner, the absence of academic programmes to give credibility to conclusions drawn and the hostile nature of the subject matter itself which seeks to obfuscate analysis. Even one of the most fundamental requirements of digital evidence, the ability to repeat and hence corroborate the findings of the expert, is open to challenge.

1.1.5 Reproducibility flaws

An established tenet of science is that hypotheses are supported by reproducible experiments (Beckett, 2010). To meet the requirements of scientific reproducibility, these hypotheses need to incorporate Popper's concept of *falsification* (Popper, 1968), the idea that a hypothesis can be proven to be false, thereby advancing one's knowledge of the subject. Typically a null hypothesis is formed and controlled tests are performed to identify the circumstances under which it can be proven to be false. The concept of reproducibility also applies to evidence prepared for criminal proceedings.

Practitioners tendering digital evidence must expect to defend their findings and disclose enough detail to enable an opposing expert to verify and possibly provide an alternative explanation for an artefact. One technique used by practitioners to mitigate against any such challenges is to compare the findings of one tool with those of another tool.

This technique is promoted as a tenet of forensic computing (Beckett, 2010). Practitioners refer to this technique as *dual-tool verification*. One forensic provider states "Dual-tool verification can confirm result integrity during analysis" (Forensic control, 2011). This is a bold claim and is open to challenge if a third tool or manual inspection of the raw data identify a discrepency. Another provider makes the less radical claim that the forensic software products EnCase and FTK "allow for a dual-tool approach for the verification of findings" (Cy4or, 2009). As before, no scientific studies or supporting evidence are cited. A third example is a freelance forensic investigator also states on his website in relation to tool validation, "I don't validate my tools - I validate my results. Generally I do this with dual tool verification" (Drinkwater, 2009). This statement is contradictory as a second tool is used to check the results of another.

This form of verification falls short of the scientific practice of verification. Even if the definition of verification is limited to a simple comparison, there is no documented record of the notion that two tools can make the same error (Beckett & Slay, 2007). This could arise,

for example, by using the same underlying Windows API call. Under these circumstances, the designs of both tools are subject to the same erroneous assumption (Sommer, 2010).

Dual-tool verification cannot confirm a result, but it can corroborate it on a statistically insignificant scale. The main benefit in applying a dual-tool approach is in identifying discrepancies in results (Turner, 2008), thereby highlighting the need for closer analysis. An example of this is in the trial of Casey Anthony [2011] who was charged with the murder of Caylee Marie Anthony in Orlando, Florida. During this trial a discrepancy was identified between two Internet history tools used to produce expert testimony. As a result of this discovery, the developer of one of the tools corroborated the tool's output by reverting to the underlying raw data and interpreting the data manually (Wilson, 2011). Although good practice, this step is not without bias on the part of the developer towards defending his code and commercial product. Ideally, an independent party unaware of the expected outcome should have undertaken this step.

The acceptance of a tool or methodology sanctioned by others is common practice in both legal and scientific circles. In judicial processes, legal precedent can be cited from prior cases where techniques have been admitted into proceedings. Scientific work advances by citing and carefully extending through hypotheses a previously established body of knowledge. The difference arises in how these precedents are determined and hence accepted.

1.1.6 Acceptance of fact

Kritzer (2009) argues scientific and legal inquiry differ in how they persuade and hence accept propositions. He argues that the scientific tenet of general acceptance and peer review is advanced through repeated attempts to falsify a hypothesis. Truth, he continues, in a scientific context is complex and elusive and can only be approached by a process of eliminating falsehoods. This differs to truth as applied within the legal context, which is revealed through the adversarial process.

In accepting a given truth, the legal enquirer values certainty, whilst the scientist values doubt and scepticism, argues Marsico (2004). He goes on to state that if justice is blind, then it will "blindly follow evidence presented as truth". Judges, he continues, whose role should be limited to evaluating the admissibility of evidence, are actually empowered to evaluate the credibility of scientific evidence. It can be argued that this power combined with the trust given to an expert's testimony has contributed to the problems identified in section 1.1.3.

The Daubert test in the USA, developed from Daubert v Merrell [1993], seeks to provide a framework to assist the judiciary in evaluating scientific evidence. Critics of this system

argue that it is flawed, as it is reliant on the existence of a "scientific community" when there is none for computer forensics (Marsico, 2004).

Beckett (2010) identifies the Latin terms *Ad populum* "appeal to the people" and *consensus gentium* "agreement of the people" to describe arguments that are flawed on the basis that they are believed by a large number of people. Citing Appel and Pollitt (2005), Beckett questions whether the consensus of the community can be trusted with a largely non-graduate educated scientific community in law enforcement.

Similar *consensus gentium* arguments are adopted by the vendor community who promote the acceptance of their software as it holds a vast market presence. In their Legal Journal (2011) Guidance Software state they have evaluated their forensic software product (named *EnCase*) against the Daubert test. In addressing the general acceptance criteria of this test, they argue that with more than 30,000 licensed users their product is generally accepted.

Van Buskirk & Liu (2006) argue that statements such as these lead to a tendency within the judicial system to presume forensic software is reliable. In their discussion, they identify issues, which they argue are indicative of reliability issues with the software. In response to this, Limongelli (2008) of Guidance Software defends the reliability of the software by citing Williford v State of Texas [2004], where it was concluded by the court that the EnCase software is reliable. However, closer examination of this case reveals that this conclusion was made on the basis of the anecdotal testimony of a single police officer and therefore not based on a generally accepted scientific process.

Limongelli goes on to cite Sanders v State [2006], where it was concluded that once the scientific reliability of a specific methodology is determined, "other courts may take judicial notice" of the result. The impact of such a decision within the jurisdiction where it applies is that this forensic product is prone to being accepted without due consideration to the impact of changes in the software version or bugs and/or errors that arise due to the environment where it is applied.

However, Carrier (2002) distinguishes between acceptance of a tool and acceptance of a procedure. He argues that in the absence of any published procedure detail, the choice of forensic tool from the limited range available will likely be based on non-procedural factors such as interface and support. He concludes therefore, that the size of the user community is not a valid measure of procedural acceptance.

Sommer (2010) identifies how, through the application of Part 33.6 of the Criminal Procedure Rules (2015), just two individuals (the opposing experts in a case) can accept novel scientific

evidence as "sufficient" for the case without committing to a more "universal" finding. This procedure is one example of the practice where for the purpose of addressing specific matters at hand, practitioners "ignore the evidence of falsification" (Saks & Faigman, 2008). A more conservative view on this is offered by Beach (2010) who suggests falsification is not treated by practitioners operating within the legal arena in the same way as scientists as the concept of truth differs between the science and legal profession. Within the bounds of a single case, truth is deemed static and not open to be re-evaluated. Denning (2005) argues this acceptance of untested theories is a wider problem within the computer science community as a whole, citing a study by Tichy (1998) that found approximately 50% of computer science papers published prior to 1995 had proposed models or hypotheses that were untested.

1.1.7 Emerging statutory requirements

In response to these miscarriages of justice, a UK Forensic Science Regulator (FSR) was appointed in 2008 with a remit to manage standards applicable to both scientific processes and individual competence (Sommer, 2011). The FSR is also responsible for developing guidelines for validating new developments.

The following year a European Union Council Framework Decision 2009/905/JHA (European Union, 2009) was passed on the subject of "accreditation of forensic service providers carrying out laboratory activities" relating to DNA and fingerprinting provision which declared that:

> "Member States shall ensure that their forensic service providers carrying out laboratory activities are accredited by a national accreditation body as complying with EN ISO/IEC 17025"

The FSR took this decision and broadened it to encompass all forensic service provision within the UK in their 'Codes of Practice and Conduct' (Forensic Science Regulator, 2011). This document aligns itself to the laboratory standard BS EN ISO/IEC17025:2005 (ISO, 2005). The UK Government has since sought to put the Codes of Practice on a statutory basis and provide investigative powers to the FSR for quality failures (Home Office, 2013). By October 2017 all digital forensic service providers (including those based within UK police forces) are required to be accredited (House of Commons, 2016).

In addition to the above, a survey for the Chatham House report entitled *Cybersecurity and the UK's Critical National Infrastructure* (Cornish, Livingstone, Clemente & Yorke, 2011)

found that the participants agreed that for cybercrime: *"*Any analyses carried out must be subject to validation for appropriateness, completeness and accuracy.*"*

1.1.8 Summary

Given the reasons for the appointment of a Forensic Science Regulator and approaching statutory standards, it can be argued that the issues identified currently undermine the trust that can be placed in findings tendered in criminal proceedings.

The production of electronic evidence therefore requires the use of reliable tools and competent operators. This research explores both areas and focuses on the trust placed in the tools used.

1.2 Research Question

The previously identified miscarriages of justice, emerging regulatory controls, ethical considerations and a desire to promote awareness of the need to test the limits of tools and their results have led to the formulation of the following question in the context of malware forensics:

> *Can a systematic basis for trusted practice be established for evaluating malware artefact detection tools used within a forensic investigation?*

In order to address this question fully, it is helpful to formulate a series of specific, more focused, sub-questions:

1. To what extent is there a case for a lack of trusted practice?
2. What are the requirements for evaluating malware artefact detection tools?
3. Do the conditions under which tools and malware operate have an effect on the ability to observe malware behaviour?
4. Are observations of malware behaviour impacted by the practitioner's choice of tool?
5. What factors can be used to evaluate the performance of the methodology and hence identify areas of improvement.

Commencing with the first question above, the definition of trusted practice used within this research is derived from the Crown Prosecution Service (CPS) (2015), who state that expert evidence must be reliable, in other words trustworthy. They go on to describe a characteristic of reliable expert evidence as having a "scientific basis". Furthermore, they also stipulate that reliable evidence should be such that it can be "reviewed by others", i.e.: is repeatable and reproducible. Hence the trusted practice in this context is deemed to be one that produces

evidence through a scientific methodology. Repeatability and reproducibility are just two of several hallmarks of the scientific method. Others include falsifiability, whereby a hypothesis is testable; controllability such that a single variable can be manipulated; and unbiased (Peisert & Bishop, 2007). These are listed in Table 1-1.

#	**Scientific method attribute**
1	Repeatability
2	Reproducible
3	Testable hypothesis
4	Controllable
5	Unbiased

Table 1-1 : Attributes of the scientific method

It is helpful therefore, to establish to what extent there is a case to answer for a lack of trusted practice. Consequently, current practice within the field of digital forensics, and more specifically within cases involving malware, is reviewed.

Furthermore, given the focus of the research question is the evaluation of software tools, then the current practice and requirements for this are also reviewed (question 2 above). To address any doubt that may be introduced as a result of operating such tools in a malware environment (thereby impacting on the trust placed in them), a study to explore the effect of different operating conditions is also undertaken (question 3 above). By subjecting different tools to such scrutiny, the practitioner will be able to compare the observations reported by different tools, thereby informing their decision in the choice of tool to use (question 4 above). Finally, the methodology identified to address the above questions should itself be subject to review and critical reflection to identify areas of improvement (question 5 above).

The principal themes of the research question are the concepts of *trusted practice*, *tool evaluation* and *forensic investigation*. Therefore, these elements inform the underlying direction of the research and together with the sub-questions above, have been used to derive a series of research goals, see Table 1-2. The chapters that address these goals are shown in the right-hand side of the table.

Goal	Sub-questions	Description	See Chapter
1	1	Determine if there is a problem with a lack of trusted practice in malware forensics	2
2	2	Identify the requirements for a solution	3
3	3,4,5	Develop a methodology for evaluating malware artefact detection tools	4

Table 1-2 : Research Goals

1.3 Research Contribution

By achieving the goals listed above, the key contributions of this research are:

1. Confirmation for case for a lack of trusted practice in the field of malware forensics
2. A set of requirements for establishing trusted practice in the use of malware artefact detection tools
3. An extensible framework to increase the level of confidence in the use of tools applied to malware analysis
4. Empirical evidence identifying the optimal execution time for a given tool when observing malware artefacts
5. Empirical evidence that the choice of tool can impact on the number of artefacts observed
6. Empirical evidence of the performance of this framework.
7. A systematic methodology for practitioners to specify operating parameters (such as how long the tool must be run for) when obtaining new or unfamiliar tools.

The following have been identified as potential beneficiaries of this research:

Who	Perceived benefits
Practitioners	• Road map of current state of research in field • Framework to empirically evaluate tools • Quantifiable means to compare tools • Ability to make informed decisions on choice of tool for malware analysis • Ability to customise test environment to evaluate a tool under different conditions
Academics	• Identification of key research groups and areas for academics seeking to undertake a further research • Supporting data for any subsequent research • Identification of risks/caveats in the research field
Criminal Justice System	• Cite gaps through authorities in field • Empirical data to validate methodology • Inform on the admissibility of evidence
Public	• Potential to reduce miscarriages of justice
Software vendors	• Identify gaps supported by authorities in field • Framework to test products against • Provide scientific underpinning to products
Forensic Regulator / Standards bodies	• Identify gaps supported by authorities in field • Empirical data to validate methodology • Methodology to inform testing/evaluation of tools

Table 1-3 : Potential beneficiaries of the research

1.4 Research strategy

The research process selected for this thesis largely follows the five-stage Action Research process (Cottrell, 2014, p. 102). Cottrell states this research method is a valid approach that can be undertaken by "practitioners into an area related to their own work". The following is a brief description of each of the steps involved in Action Research as employed in this research.

a. **identifying** a research question (diagnosing). This process was largely exploratory in nature, where the research problem was articulated from gaps in available literature, regulatory requirements, presentations, blogs and discussions with peers within the digital forensic community and criminal justice system, etc. This step sought to answer research sub-questions 1 and 2 (see Section 1.2).
b. **developing** an action plan (action planning). This initially involved the development of a realistic timeline for each task that would eventually lead to answering the research questions identified in (a) above. Much of the development stage concerned identifying the components of the framework and determining how they would interoperate (see Sections 4.1 and 4.2).
c. **implementing** the plan (action taking). This stage involved the instantiation of the framework through the development of the code that forms the components identified in (b). In parallel to the code development, links were established to the malware source and online malware analysis providers to determine how the code would interface to their systems (see section 4.3).
d. **gathering and analysing the data** (observing). Data in (c) above was collected and analysed to provide empirical evidence to test the research hypothesis. Steps (b), (c), and (d) all contribute to providing an insight into the solutions of research sub-questions 3, 4 and 5 (see Section 1.2).
e. **reflecting** on the findings of the investigation (evaluating). The results of step (d) were used to further draw a conclusion on the significance of the contribution this thesis makes to tool evaluation in a malware forensic context.

1.5 Thesis structure

This thesis is structured as follows:

Chapter 2 surveys related literature and identifies the relatively small amount of groundwork that has begun to emerge to establish digital forensics on a scientific footing. However, there is little empirical research to underpin malware forensic practice, which is based upon anecdotal and ad-hoc processes. The chapter closes with a review of tool evaluation methods.

Chapter 3 examines the gap between the state of the art in malware forensic practice and the technical, legal and regulatory requirements of such a process operating within the Criminal Justice system of the UK. The chapter closes with a series of requirements reflecting the disparity between current and required practice.

Chapter 4 opens with the aims of the design and proceeds to identify the main components of the framework to address the previously identified gap. The latter half of the chapter examines the implementation of the framework and proposes a testing and analysis strategy.

Chapter 5 reports on the results of a series of experiments conducted using the implemented framework. An analysis and discussion section follows where it is found that the both the length of execution time and choice of tools impacts on the number of artefacts observed.

Chapter 6 evaluates the framework from a number of different perspectives. Early on in the chapter, an evaluation against the requirements and aims is undertaken. Performance is also evaluated, looking at the speed and resource utilisation. How well the framework addresses the research question is also evaluated. The chapter closes by identifying the limitations of the framework and then leading into further work proposals.

Chapter 7 presents the conclusions of the research and in doing so, offers a critique of the thesis itself. The contributions are identified before the chapter closes with a summary of the proposed further work.

Chapter 2 Literature search

Chapter 1 outlined a number of issues that undermine the trust placed in forensic evidence. A greater confidence in the tools used in practice is required as the discipline moves towards increased regulation (see section 1.1.7).

In order to understand the context of where these tools are used and help identify the steps needed to address the trust issue, this background chapter is divided into three principal sections. The first of these provides a review of digital forensic practice and identifies a relatively small amount of groundwork that has begun to emerge to establish digital forensics on a scientific footing. The second section argues that there is even less empirical research to underpin malware forensic practice, which is based upon anecdotal and ad-hoc processes. The chapter closes with a review of tool evaluation methods.

2.1 Digital Forensic practice workflow

Digital forensic practice is a relatively young field and like any fledging field of study, it has attracted a number of attempts to model it. Pollitt (2007) provides a useful summary of several early process models. Among these the 2001 Digital Forensics Research Workshop (DFRWS) was one of the first significant initiatives to define the discipline by academics and practitioners alike. With over 300 citations in 15 years (Google, 2016a) it resulted in a six-stage process describing the entire lifecycle of a computer forensic investigation (Palmer, 2001). Carrier's abstraction model (Carrier, 2003) is also widely cited (Google, 2016b); it uses abstraction layers to form a model for digital data being examined during forensic analysis at a high level. Alongside the input and output data of each layer, Carrier argues there is also a 'Rule Set' that defines the interpretation of the layer together with a 'Margin of Error', see Figure 2-1. Carrier provides an example of binary input data that has an ASCII mapping rule set applied to it. The output of this would be the alphanumeric representation of the data. This, he argues, could then be fed into another layer. If the data were the contents of an HTML document, then the alphanumeric characters would become the input along with the HTML specification as a rule set to produce a formatted document as an output. Carrier (2006) subsequently used abstraction layers to model a digital forensic investigation using finite state machine theory.

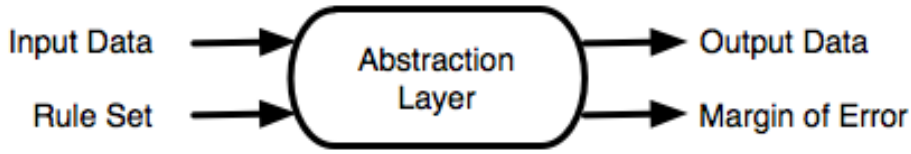

Figure 2-1 : Abstraction layer inputs and outputs, adapted from Carrier (2003)

Carrier's model has been criticised by Flandrin *et al.* (2014) as being too complex to implement as an "extensive digital forensics knowledge of the internals of the tool" would be needed but most tools used are closed source.

More recently, Raghaven (2012) has provided a series of taxonomies that summarise the field from different perspectives, including attempts at modelling the forensic process.

Kaur & Kaur (2012) suggest that despite the variety of models proposed, many of them are ad-hoc and hence have not been adopted by the practicing community. However, they do not elaborate on why they deem them to be ad-hoc. Brown (2010) suggests the lack of adoption by the community may be the result of the need for practitioners to adapt their workflow to be "general enough to be useful in an array of situations" and the fear of being challenged in court for not following a Standard Operating procedure (SOP). Vincze (2016) describes how this remains an open problem by citing Casey (2011a) and Pollitt (2010), pointing out that "after decades of discussion, the debate continues". This echoes the view that to date that an adopted comprehensive digital investigation process model simply "does not exist" (Montasari, Peltola & Evans, 2015).

Despite this claim, Kent *et al.* (2006) previously published a report on behalf of the National Institute of Standards and Technology (NIST) in which a four-stage process model was presented. The fact that a standards body has produced the model may bridge the gap between the academic and practitioner community and therefore increase the likelihood of it being adopted in practice. It is perhaps for this reason that Zareen *et al.* (2013) give more emphasis to this model in terms of coverage over others and describes this model as an "established" procedure "accepted the world over", see Figure 2-2.

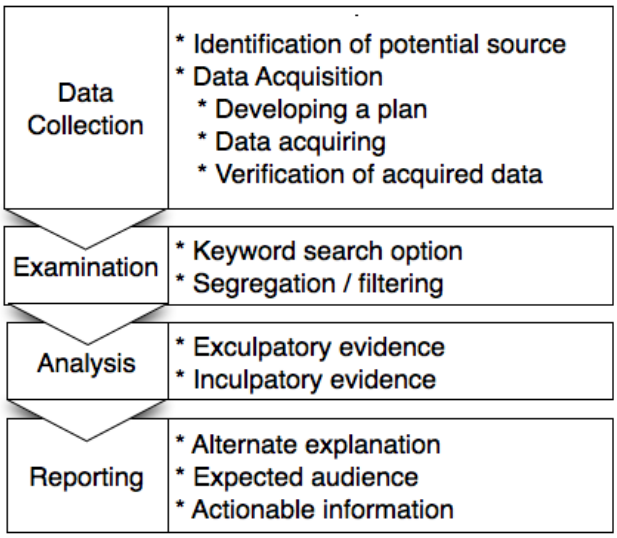

Figure 2-2 : NIST Digital Forensic procedure, adapted from Zareen *et al.* (2013)

In the United Kingdom, there is a move to adopt the ISO 17025 Standard (ISO, 2005), which is incorporated into the Forensic Science Regulator's Codes of Practice and Conduct (2016). Given the lack of consensus within the community it is not a surprise to note that the Regulator has laid down no formal model or recommended methodology for the discipline. The European Anti-Fraud Office (OLAF) published their updated standard operating procedures for conducting digital forensics investigations in February 2016 (OLAF, 2016). These guidelines are aimed at OLAF staff and agencies operating on their behalf. Furthermore, their focus is somewhat high level and designed to help ensure compliance with data protection provisions in the context of digital forensic operations. The only other guidelines applicable to practice in the UK are the ACPO Good Practice Guide for Digital Evidence (Williams, 2012). This document provides a recommended methodology for the acquisition of digital evidence, but not for the process as a whole.

Kipling (2012) again echoes the lack of a standardised methodology for conducting a digital forensic investigation. However, unlike others, she extends this observation to the absence of a methodology for "searching for malware". She argues that existing methodologies focus on the defendant's "actions on the computer to prove intent", i.e.: *mens rea*. This corresponds to the Analysis stage of the NIST procedure (see Figure 2-2) where the overall objective is to label evidence as either exculpatory or inculpatory. Kipling points out that methodologies that focus on user activity are too limited in their scope where malware is involved. There is a requirement to differentiate the actions of a user with those of malware and to take into account anti-forensic measures such as tampering with file timestamps that may have occurred. Approaches to this requirement will be explored in the next section.

2.2 Forensic analysis in a malware environment

Malware analysis is typically undertaken by security researchers and generally seeks to answer one or more of four questions: can it be detected (Huda et al., 2016); can it be classified (Daly & Burns, 2010); can its behaviour be understood enough to comprehend its objective(s) (Zolkipli & Jantan, 2011); or can it be neutralised (Morales, Sandhu & Xu, 2010). The strategy taken to analyse malware is largely divided into static or dynamic analysis (Egele, Scholte, Kirda & Kruegel, 2012), whilst others adopt a hybrid approach of both (Shijo & Salim, 2015). Egele *et al.* also identify three platforms for implementing such analysis; namely *bare metal*, *virtualisation* and *emulation*, see Figure 2-3.

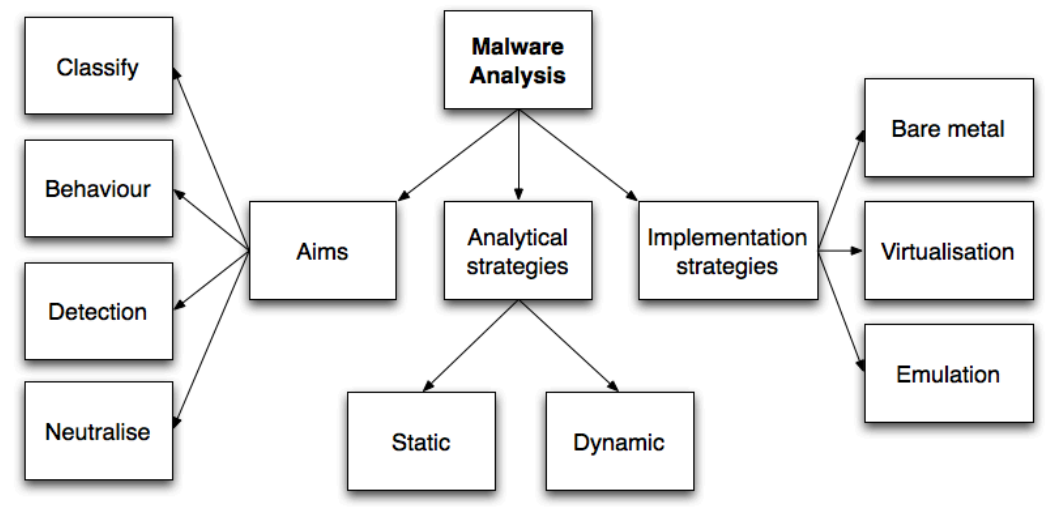

Figure 2-3 : Malware analysis - Aims & strategies

A *bare metal* platform is one where the operating system on which the analysis is to be performed is installed on physical machine (like a regular desktop computer). Malware analysis performed on a bare metal platform is the most authentic, as it most closely reflects what would happen on a computer, once infected with malware.

On a *virtualisation* platform the privileged state of the physical machine (akin to *root* permissions access to hardware resources) is not directly accessible to a virtual machine running on the platform. A Virtual Machine Manager (VMM) manages access to these resources. Furthermore, both the host and guest machines must have same underlying instruction set architecture, such as the Intel based x86 or x64 instruction set (Intel, 2016).

An *emulation* platform allows for the host and guest architectures to be different. This means that a guest computer that uses a different underlying instruction set architecture, such as an older Apple Mac based PowerPC , could be hosted by a computer running the Windows

operating system. Also, the host machine has full control over what the guest can see, so the analysis tools can remain undetected from the malware. However, some malware can detect the side-effects of emulation, such as the features of an imperfectly emulated CPU (Egele, Scholte, Kirda & Kruegel, 2012). These three implementation strategies are covered in more detail in section 2.2.4.

From a forensic investigation perspective, the review of the literature has uncovered little published material documenting the procedure for conducting a malware forensic investigation or indeed for evaluating the tools to do so. The use of malware forensics is cited by Kim *et al.* (2014) who present a model to investigate fraud using "malware forensic" techniques. General acceptance of the term is demonstrated by the fact that it is emerging within other sectors, such as education. Techniques such as *gamification* are being deployed to teach malware forensics as part of a wider digital forensics course (Pan, Schwartz & Mishra, 2015). Shosha *et al.* (2013) present an automated approach to reconstruct forensic actions from low-level code and determine a suspect program's behaviour using a state analysis approach. Their approach uses finite state machine theory and claims to be 80% effective at identifying the actions of malware. However, their evaluation of this approach is unclear; but seems to be reliant on the ability to reverse engineer the malware to determine how closely the predicted actions follow the underlying code. Furthermore, no account for the changeable nature of malware appears to have been considered.

This changeable nature issue is address by Provataki and Katos (2013) whose *malware forensics framework* extends the functionality of the Cuckoo sandbox (Cuckoo Foundation, 2016) and provides a means to execute malware multiple times across different environments to gather an overall picture of it's *modus operandi*. The framework is designed to provide damage assessment following a malware breach and includes empirical results. However, its purpose is to evaluate malware behavior and not to evaluate the tools used to study such behavior.

Published strategies for performing malware analysis in support of law enforcement are few and far between. Ianelli *et al.* (2007) offer a discussion on the topic and suggest that the presence of malware can be addressed by examination of the network traffic logs. However, this suggestion assumes that such logs are more likely to be found in a corporate than domestic environment. Hence, a suspect accused of committing an offence via their home router will typically have far fewer logs and/or detail to assist their defense than in a commercial environment with what would likely be more sophisticated logging available.

Malin *et al.* (2008) present one of the few books on malware forensics, more recently split into separate Windows (2012) and Linux (2013) editions. Carvey (2012) also provides some coverage of the topic across two chapters from an investigative perspective, as part of a more general digital forensics discussion. Each of these texts presents a collection of tools and techniques to address various aspect of analysis, but none attempt to develop and evaluate a general-purpose framework for malware analysis.

Notwithstanding this lack of a framework, Malin *et al.* does suggest five broad phases to a forensic investigation involving malware that is clearly aimed at the practitioner.

- Phase 1 : Forensic preservation and examination of volatile data
- Phase 2 : Examination of memory
- Phase 3 : Forensic Analysis: Examination of hard drives
- Phase 4 : Static analysis of malware
- Phase 5 : Dynamic analysis of malware

Malin *et al.* take the view that, "within each of these phases formalized methodologies and goals are emphasized". Taking Phase 4 as an example they present a file profiling methodology as a static analysis approach to studying malware, see Figure 2-4. However, they offer no provenance on the methodology, no evaluation against any alternatives, nor any argument why this particular approach was selected.

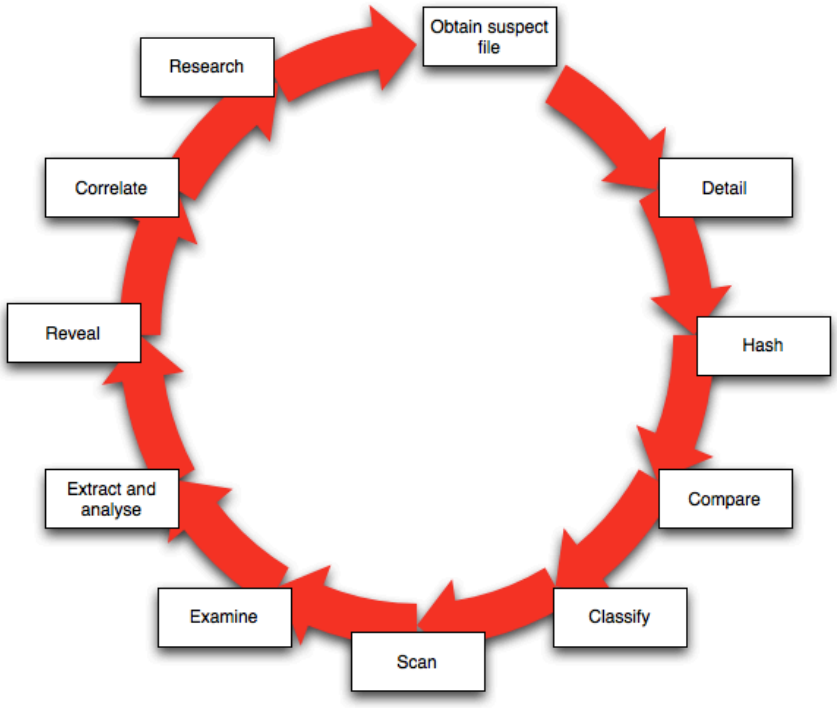

Figure 2-4 : Steps in the File Profiling Process, adapted from Malin *et al.* (2008)

Despite these shortcomings (and in the absence of viable alternatives), this structure was used as a starting point to divide up the discipline into different topics, as applied to evaluating the tools used in a forensic investigation involving malware. The remainder of this section has been divided into subsections to consider the viability of each of these phases to the aims of the Research Question in section 1.2. Phases 1 and 2 are quickly dismissed as not being viable, given the challenges and resources available within the context of the PhD. A crowded research space, such as Farely (2015) who presents a method to perform automated forensic analysis of malware and Kim *et al.* (2014) who use automated malware forensic techniques to detect financial transaction anomalies, dismisses Phase 3, whilst Phase 4 is similarly dismissed on the grounds that such tools are unlikely to be used owning to the additional skills (such as assembly language and reverse engineering) needed to interpret their results. A lack of support for a command-line interface and hence scripting capability also led to issues with automation of such tools. This contributed to the decision to discount implementing this phase in the approach to this PhD. The final phase proposed by Malin *et al.* is a more viable option for this PhD and is therefore given greater coverage in section 2.2.4.

2.2.1 Phases 1 and 2: Preservation, volatile data and memory

Arguably there is some overlap between some of the phases of this approach. For example, elements of what you might expect to be in the preservation of volatile data (such as recording network port activity in Phase 1) would also be present in the dynamic analysis of malware, where the malware file is executed and studied (Phase 5). Similarly, purists may argue that an examination of memory (Phase 2) should encompass both volatile data (secured in Phase 1) and paged memory, hibernated code or even crash dumps, all of which are to be found on storage media (Phase 3).

Given these overlaps and the ephemeral nature of RAM data, the first two phases of the approach identified by Malin *et al.* have been excluded, on the basis that it is more challenging evaluating tools to meet the aims of the Research Question in section 1.2. In addition, the nature of RAM acquisition brings challenges to the repeatability and reproducibility of its acquisition, thereby falling short of a scientific method, as defined in this research (see Table 1-1). Furthermore, Malin *et al.* present the first phase very much from an incident response (as opposed to a forensic investigation) perspective.

The remaining three phases are therefore all plausible candidates for the focus of this research. What follows if a brief review of each of these.

2.2.2 Phase 3: Forensic Analysis Examination of hard drives

The third phase is aligned to a conventional digital forensic examination workflow and according to Malin *et al.,* is concerned with the use of more established forensic analysis tools, such as EnCase. Thus, temporal analysis of artefacts in the form of timelines can be generated.

Malin *et al.* present an anecdotal methodology for this phase (see Figure 2-5) that they state "provides the greatest chance of finding the majority of evidence relating to malware on a computer". With no supporting evidence to back up this claim, it is presented very much as practitioner guidelines, rather than a scientifically tested and evaluated process. Furthermore, no discussion on the order of the process steps shown in Figure 2-5 is presented, leaving the reader uncertain of any dependencies and thus the impact of changing this order. Some steps (such as searching for known malware) would need to take place before others (such as inspecting an executable), whilst steps such as reviewing user accounts would not.

Implementation

The steps presented by Malin *et al.* in Figure 2-5 can largely be completed using dedicated forensic analysis software, such as EnCase or SleuthKit. As Malin *et al.* point out, there are steps in this phase where the practitioner is obligated to use tools not designed for forensic use:

"The increasing use of malware to commit and conceal crimes is compelling more digital investigators to make use of malware analysis techniques and tools that were previously the domain of antivirus vendors and security researchers".

This practice of filling the void left by the dedicated forensic tools by using tools not designed for use in a forensic context is also recognised by Beckett (2007). It presents an opportunity to challenge the integrity of the evidence produced using such tools.

Tool evaluation opportunities

To consider the inclusion of tools from this phase to be evaluated as part of this research, the ability to apply a high degree of automation to use of the tool was considered. To form generalisations from statistical analysis, a high level of automation is required to gather sufficient quantities of data. Some tools/tasks are difficult to automate, as they require a degree of human interpretation. For example, due to the transitional portfolio of software that is available to install, any scripted process to "Review Installed Programs" as indicated by Malin *et al.* in Figure 2-5 is likely to be quickly out of date. Furthermore, attempts to evaluate tools that "Inspect Executables" face similar challenges in that where the files being inspected are malicious they will nearly always be obfuscated and designed to misdirect any analysis. Bayer *et al.* (2006) point out that malware authors deliberately write their code to "thwart both the disassembly and code analysis"; others have presented similar views (Wagener, Dulaunoy & Engel, 2008) (Sikorski & Honig, 2012).

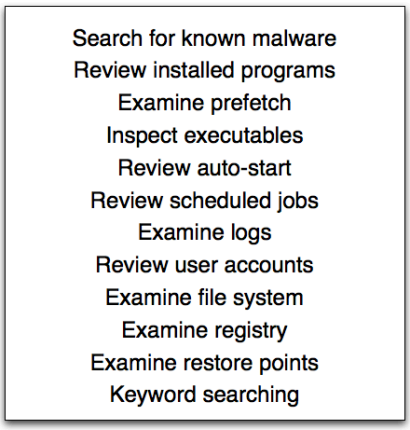

Figure 2-5 : Uncovering malware trace evidence, adapted from Malin *et al.* (2008)

Furthermore, referring to Figure 2-5, the tasks to automate the various review steps, such as scheduled jobs or log files and applications set to start automatically on boot, can all easily be scripted within tools such as EnCase. However, it is much harder to automate any interpretation of the results to differentiate between the suspicious and the benign.

Consideration was given to evaluating anti-malware scanners, which could conceivably be easily automated. This idea was abandoned due to the level of work underway by other groups, such as Harley (2012) who discusses standards for testing anti-malware products; Košinár *et al.* (2010) who discuss an anti-malware testing methodology; as well as Shijo and Salim (2015) who present a combination of both static and dynamic analysis techniques to detect malware. Harley (2012) also takes the view that not all of these groups are proficient in scientific testing methodologies. The testing itself, he contends, is largely carried out and/or interpreted for wider dissemination by non-specialists. Potter and Day (2009) present a discussion on the effectiveness of anti-malware testing (but provide no empirical data to support their position). This is in contrast to Sukwong *et al.* (2011) whose empirical study of six anti-malware products applied two stages of evaluation: file signature and behavior analysis. Corregedor and Von Solms (2012) meanwhile, compare nine commercial anti-malware products, which they evaluate against a series of requirements as part of a framework. They conclude all nine products tested have several vulnerabilities that need to be addressed. Ford and Carvalho (2014) share the concern for the lack of science in the testing anti-malware products by stating this deficiency actually "harms the industry".

In conclusion, the best opportunities for tool evaluation in this phase are limited to anti-malware tools, but this area is somewhat crowded with a number of active research groups. Therefore, attention is turned to the fourth phase, the static analysis of malware.

2.2.3 Phase 4: Static analysis of malware

Malin *et al.* describe the fourth phase as being the static analysis of malware. The steps they identify for this phase are illustrated in Figure 2-4. As previously stated, they do not provide any evaluation of these steps nor suggest any alternatives. Similar criticisms can be levied against the process proposed by Elisan (2015) who defines static analysis as the process of collecting information from the file while it is not running. From an investigative perspective, this is largely a metadata analysis that is akin to a forensic examination of the paper, ink, fibres and postmark of a physical suspect letter. Elisan provides a breakdown of the "basic steps and techniques" that are needed to conduct an "effective static analysis":

- ID assignment
- File type identification
- Antivirus detection
- Protective mechanisms identification
- Portable Executable (PE) structure verification
- Strings analysis
- Static code analysis

As with Malin *et al.,* there is a lack of clarity on any dependencies that may (or may not) be present in this process. A detailed comparison of these two approaches is outside the scope of this literature review, since the focus is on identifying the opportunities to evaluate the tools involved.

Implementation

A study by Namanya *et al.* (2015) evaluated three static analysis tools, namely *Mastiff*, *Pyew* and *PEframe*. They identify these as being the "most popular open source malware static analysis tools", though no supporting data or citation is provided to back up this claim.

The framework offered by Kipling (2012) provides a methodology to determine if malware is or was on a system, from "Indicators of Compromise" (i.e.: artefacts) left behind. Although Kipling explicitly states the approach is aimed only at "finding malware", many of the tools she cites (listed in Appendix C of her dissertation) can be used to inform the investigator about the behaviour and/or intentions of the malware.

Tool evaluation opportunities

Static analysis is frequently hindered by the use of packers that encrypt a malware binary (Raphel & Vinod, 2015). Hence, one of the first steps required during static analysis is to unpack the malware to produce a 'plain text' version of the file that can then be analysed. This can be done manually, but requires skill (and patience) in low-level assembly language, as well as a familiarity with the Portable Execution file format.

Tools are available to simplify this process, but as Lyda and Hamrock (2006) point out, unpacking tools may inadvertently execute the packed code (so precautions need to be made in the event the file is malicious) and that many of the unpacking tools are poorly written and break due to bugs and errors.

Furthermore, there are some forms of malware that do not unpack completely as an anti-forensic measure (Royal, Halpin, Dagon, Edmonds, et al., 2006). Coogan *et al.* (2009) present a solution for unpacking code that has been secured using both custom and commercially available packers without executing the malware.

In addition, it could be argued that findings, such the inclusion of network related dynamic link libraries (DLLs) could simply be circumstantial and may not mean that the file under analysis is or ever was capable of contacting the Internet, for example. Malin *et al.* (2008) take this one step further and warn that string information, for example, can often be planted to "throw digital investigators off track". Knowledge of these caveats have the potential to undermine a case that advocates that malware was the cause of the *Actus Reus* during a trial.

Regardless of the authenticity of the data examined, Provataki and Katos (2013) provide an important observation, particularly applicable to a forensic practitioner perspective. They point out that while static analysis as a process has the potential of completely uncovering a malware's inner structure and characteristics; it also requires extensive expertise, manual effort and time to perform. This, they argue, might not always be feasible to perform due to extremely sophisticated obfuscation methods and multilayered packing mechanisms embedded within the malicious code. As a consequence, the likelihood of this approach being used by Law Enforcement is arguably, quite small. Consequently, the requirement to evaluate such tools is also small. Furthermore, the required expertise and manual effort reported by Provataki and Katos (2013) inhibit the ability to automate the process and determine their reliability. Executing the malware overcomes many of these issues in that it is unpacked and will typically (but not always) create artefacts on the disk and/or network that can be observed.

2.2.4 Phase 5: Dynamic analysis of malware

Malin *et al.* describe their fifth phase as being the dynamic analysis of malware. They define this as "executing the code and monitoring its behavior, including its interaction and effect on the host system". The definition offered by Egele *et al.* (2012) extends the interpretation of Malin *et al.* from monitoring to the act of verification of actions as they describe dynamic analysis as being "techniques that execute a sample and verify the actions this sample performs in practice".

The act of executing the malware offers significant advantages over static malware analysis. One such advantage is the increase in the speed of analysis (Provataki & Katos, 2013). As argued by Seifert *et al.* (2007), this is likely to be as a result of how dynamic analysis simplifies and automates the analysis process. Ross (2010) adds that dynamic malware analysis does not require specialist skills such as "an extensive understanding of assembler". Elisan (2015) also suggests that such an approach "reveals most of its functionalities".

In contrast to Elisan, Provataki and Katos (2013) argue that the behaviour of a malware binary may vary subject to the conditions under which it was run and so only a portion of the malware's behavior may be exhibited. Sikorski and Honig (2012) pick up on this point declaring that "not all code paths may execute" when running malware.

Implementation

Dynamic malware analysis can be implemented using one of two broad approaches, namely transition and state based logging (Liao & Langweg, 2014), see Figure 2-6. For this discussion, they will be referred to as Process states and Snapshots, respectively.

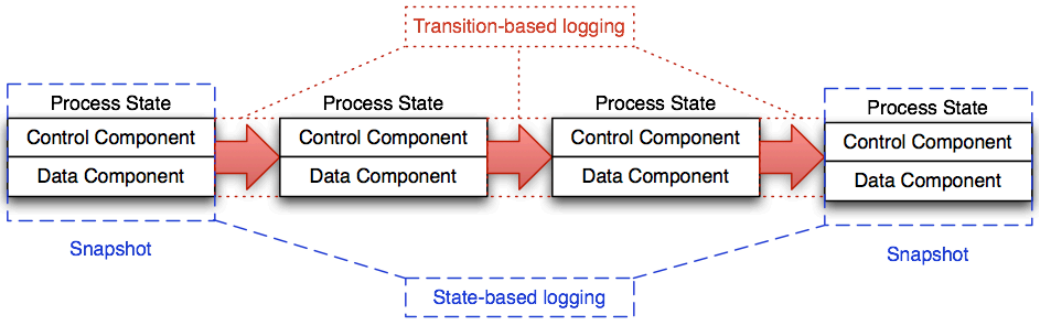

Figure 2-6 : Transition vs State logging, adapted from Liao and Langweg (2014)

State-based logging periodically samples the state of the system, gathering a much less granular record of changes than possible with transition-based logging. Transition-based

logging monitors for specific events which are typically more easily recorded. Both approaches require deciding in advance on the level of granularity to be recorded.

Liao & Langweg point out that transition based logging has an advantage over state based logging in that if an incident occurs between two snapshots then the trace information for this incident will not be missed, as each change would be recorded as a new process state. An example of this would be a file that is created after the initial snapshot, which is then deleted before the second snapshot is created.

Malin *et al.* refers to these two approaches as Active (Transition based) and Passive (State based) monitoring, see Figure 2-7.

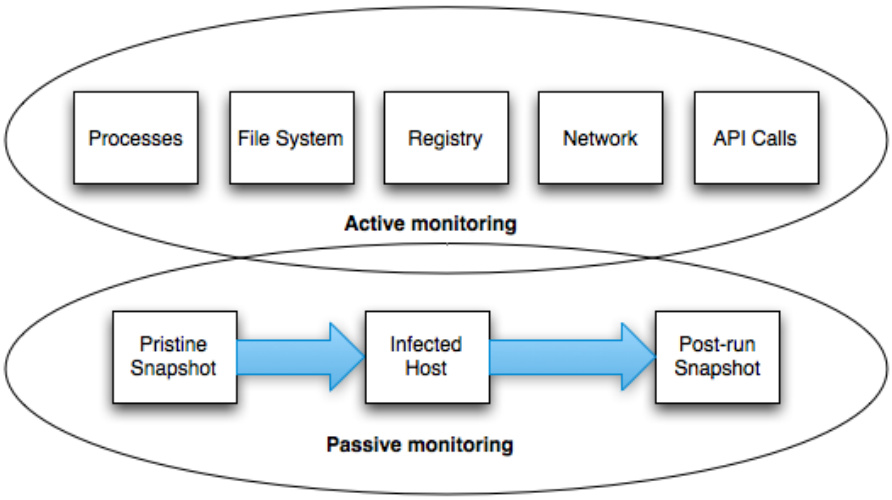

Figure 2-7 : Active vs Passive monitoring, adapted from Malin *et al.* (2008)

These two approaches follow on from a series of guidelines that Malin *et al.* recommend be followed as part of a dynamic malware process:

1. Establishing the Environment Baseline
2. Pre-execution Preparation
3. Executing the Malicious Code Specimen
4. System and Network Monitoring
5. Environment Emulation and Adjustment
6. Process Spying
7. Defeating Obfuscation
8. Decompiling
9. Advanced PE Analysis
10. Interacting with and Manipulating the Malware Specimen
11. Exploring and Verifying Specimen Functionality and Purpose
12. Event Reconstruction and Artefact Review

Thus the tools that would apply here would be largely designed to monitor changes to a computer system, either in real-time or as a comparison of before and after snapshot (Egele, Scholte, Kirda & Kruegel, 2012).

Unlike previous phases, tools used for dynamic malware analysis also require an environment within which to operate. Referring to Figure 2-3, the implementation strategies available for this are *bare metal*, *virtualisation* and *emulation*.

Bare metal implementations such as Tomlin's 'Litterbox' cited by Willems *et al.* (2007) run the malware directly on physical hardware to achieve the most realistic conditions possible. However, such approaches are resource intensive and parallel processing is limited to the number of physical machines available. Furthermore, the throughput of analysis is limited to the time it takes to restore the system to the pre-infection state (Grégio, Afonso, Filho, Geus, et al., 2015). An interesting study named *BareBox* (Kirat, Vigna & Kruegel, 2011) seeks to address these limitations by restoring the entire physical memory of the target operating system with a clean one without rebooting the system. However, according to Grégio *et al.* this approach fails to detect more privileged actions such as drivers loading. A more efficient approach is to use virtualisation instead, as it requires fewer physical machines and can achieve greater scalability and throughput. Furthermore, the time to reset a machine for a subsequent analysis is shorter.

Virtualization is a process that involves simulating parts of a computer's hardware to a point where a guest operating system can run unmodified. Most operations still occur on the real hardware for efficiency reasons. However, both the guest and host operating systems must share the same architecture, with the host providing any required backwards compatibility, such as a 32-bit operating system hosted by a 64-bit operating system of the same instruction set family (Boley, 2014). As Boley points out, this differs to emulation where the CPU, memory and other devices are all emulated in software and have no direct access to the host's hardware.

Until recently, a criticism of performing malware analysis on a virtual machine (VM) was that some malware is *VM aware* and so would not run in the same way as it would on a physical machine Krister (2009). Martignoni *et al.* (2009) warned that as a result of using "synthetic" environments results would "very likely" be incomplete. Chen *et al.* (2008) exploited this by implementing fake artefacts on a virtualised guest, thereby protecting it.

However, following an empirical study comprising of 200,000 malware samples taken from 2012 to 2014, Wueest (2015) argues that only a small number of these cases detected it was running within a VM.

This section has explored the process of dynamic malware analysis. When applied to a digital forensic investigation, the Research Question (section 1.2) seeks to establish a methodical approach to elevating the level of trust placed in the tools used to perform this analysis. Hence a means to evaluate the tools is required. The next section will consider the elements that make up the tool evaluation process.

2.3 Tool evaluation

Given that forensic practitioners can be called to give evidence under oath of their findings and their interpretation of such findings, it is not unusual for practitioners to seek to evaluate their tools before trusting them. However, recall from section 1.1 a variety of problems impacting on this trust, including the ad-hoc nature of this process and the need to more formally evaluate tools to meet the requirements of the FSR.

Therefore, this section looks at the definition of tool evaluation, taking into account the overarching goals of the research. It opens with a high level tour of the literature to highlight the main approaches and give context to the topic. Following this, the criteria that can be used to determine what successful evaluation 'looks like' is discussed. The benefits, risks and challenges of such an activity are also included, as well as a discussion on who is best placed to perform the evaluation. Finally, the section concludes with a more detailed review of evaluation methodologies proposed by others.

2.3.1 What is tool evaluation?

The evaluation of software is an established process that is embodied in two fundamental scientific concepts in software engineering referred to as "Validation & Verification" (V&V), succinctly defined by Boehm (1989) to mean:

- Verification: Are we building the product right?
- Validation: Are we building the right product?

It has become common practice for organisations to integrate these models into the normal working processes as part of their software development life cycle. Beckett (2010) cites the US Department of Health Food and Drug Administration (FDA, 2002) and the Independent verification and validation facility for NASA (Asbury, 2015) as just two examples.

Organisations that develop software will typically have documented procedures in place to maintain quality standards within the software development lifecycle from unit testing through to product testing. Beckett goes on to argue that despite an increase in the use of programming methods (such as agile and eXtreme programming) that incorporate testing as

part of the development lifecycle, no evidence of a published testing model could be identified for any forensic software product. This follows an earlier publication where Beckett & Slay (2007) argued that the evaluation of tools is "widely undocumented, and not proven publicly, except through rhetoric and hearsay on the bulletin boards of individual tool developers". This is in conflict with the repeatability and reproducibility attributes of the scientific method (see Table 1-1). They go on to suggest that one reason for this might be the difficulty, cost and resource challenges this poses.

Flandrin *et al.* (2014) agree with this lack of published material on digital forensic tool evaluation, arguing that most of what is published tends to focus on methodologies rather than the tools used. Furthermore, as a result of the literature review, no material at all has been identified regarding the evaluation of tools used to investigate malware.

The Oxford English Dictionary (2016a) defines the term 'evaluate' as, "To 'reckon up', ascertain the amount of; to express in terms of something already known". The American centric Merriam-Webster dictionary (2017) defines the term as, "to determine or fix the value of; to determine the significance, worth, or condition of usually by careful appraisal and study". Both of these definitions make reference to quantifying a value.

However, simply knowing the value of something related to malware artefact detection tools may be insufficient for their use in a legal context. Garfinkel *et al.* (2009) points out that in the USA the legal test that determines the acceptability of a forensic tool is that it must "reflect the data accurately". Rule 1001(3) of the US Federal Rules of Evidence sets out this requirement for accuracy, but does not offer a definition for the term.

Until its repeal in 1999 by the Youth Justice and Criminal Evidence Act (1999) the United Kingdom took a similar stance using Section 69 of the Police and Criminal Evidence Act (1984) that stipulated that electronic evidence will not be admissible if it is "inaccurate". This repeal effectively places digital evidence on the same footing as any other form of evidence, meaning it is presumed to be valid and thus may be admitted unless evidence to the contrary is provided (Lloyd, 2014).

However, by October 2017 all practitioners operating within the Criminal Justice System in the United Kingdom will need to abide by the Forensic Science Regulator's 'Codes of Practice and Conduct' (House of Commons, 2013). These codes include the requirement that "measurement based methods" must be "accurate". This requirement supports the argument of Flandrin *et al.* (2014) that it is the method and not the tool that must be accurate. Accuracy is defined in the same document as:

> *"The closeness of agreement between the mean of a set of results or an individual result and the value that is accepted as the true or correct value for the quantity measured."*

Others argue that accuracy is not the only means by which to evaluate digital forensic tools. Ayers (2009) includes accuracy amongst seven metrics that are available to measure the "efficacy and performance" of digital forensic tools:

- **Absolute speed** the time required by the tool to complete a task.
- **Relative speed** compares the average processing evidence rate against the rate to read data from the original media.
- **Accuracy** is the proportion of correct results.
- **Completeness** represents the proportion of evidence found from the pool of evidence available in the forensics image.
- **Reliability** measures how often the tool is likely to fail during an investigation.
- **Auditability** defines if the results are fully auditable.
- **Repeatability** measures the proportion of tests where the process employed was exactly as specified.

Aside from the connotations of influence and bias in the term "efficacy", there are ambiguities in these definitions provided by Ayers (2009). For example, it is unclear what is meant by "fail" under the term "Reliability" whilst "Repeatability" is defined in terms of the similarity of processes, rather than the results. In addition, neither of the definitions for these two terms aligns to the definitions based on the scientific method recognised by the CPS (2015), making them unsuitabile for this research. Furthermore, Flandrin *et al.* (2014) point to the lack of clarity on the meaning of "correct results" under the metric "Accuracy".

Liao & Langweg (2014) offer a review of process activity tracking systems from a forensic analysis and forensic readiness perspective. Classifying the tools reviewed in terms of their implementation strategy (e.g.: kernel vs. user space monitoring), they claim to evaluate the tools in terms of soundness, completeness, timeliness, and cost of process activity tracking. Unfortunately, there is little detail in terms of the results for such analysis included in their review.

One of the more quantifiable attempts to evaluate software tools is offered by Saleem *et al.* (2012) who use ratios of completeness of expected and observed artefacts coupled with statistical analysis to evaluate mobile phone acquisition software. This approach is

problematic when applied to malware artefact analysis, as the random nature of malware can mean that more artefacts are observed than were expected. This results in ratios (used by Saleem *et al.* to calculate p-values) greater than 1.

Where tools are used in a live environment, additional metrics related to the impact such a tool has on the systems, such as memory footprint, locally changed files, network or registry keys should also be considered (Sutherland, Evans, Tryfonas & Blyth, 2008). Lempereur *et al.* (2010) argue that the metrics considered by Sutherland *et al.* are "inconsequential" and suggest the memory of two virtual machines run in parallel be compared instead, where one virtual machine is monitored and one is not.

Boehm's V&V concepts (stated above) were formally incorporated into the IEEE 1012-2004 standard (IEEE, 2005) and subsequently the ISO 17025 (ISO, 2005, p. 17025). Evaluation by V&V can be categorized into two groups: White box testing and Black box testing (Liang, 2010). White box testing is appropriate if the individual conducting the test has access to the source code. However, as Liang points out this is not the case for mainstream forensic software tools, which are closed source.

To overcome this problem, Black box testing can be employed to test the functionality of such products against expected outcomes. The Computer Forensics Tool Testing (CFTT) programme (NIST, 2003b) have developed a series of specifications containing assertions of functionality that can be used to evaluate both software and hardware products.

Forensic processes can be complex in nature and not easy to automate or otherwise define in a prescriptive enough way (Ayers, 2009). Consequently, the processes defined by the CFTT project are quite limited in scope, with much of the work focusing on acquisition methods. It is perhaps of little surprise then that much of the literature reflects this with independent testing of EnCase imaging (Byers & Shahmehri, 2009), a comparison of the imaging functionality of three tools (Cusack & Liang, 2011) and mobile phone acquisition (Kubi, Saleem & Popov, 2011).

A more concerted effort to define the functionality of digital forensic tools is offered by the Defence and Systems Institute at the University of South Wales in a series of papers by Wilsdon and Slay (2005, 2006), Beckett & Slay (2007), Guo *et al.* (2009) and Guo & Slay (2010c, 2010a, 2010d, 2010b). Through these publications, this group has attempted to map out the functionality of basic forensic processes, such as keyword searching. However, not one of these publications actually implements their proposed framework to evaluate any tools.

What seems clear here is that there is no clear consensus in the literature on what *tool evaluation* is in a digital forensic context. Furthermore, no mention at all has been found to defining the concept in a malware analysis context. A way forward from this dilemma might be to consider what criteria a digital forensic tool, in particular one used for malware analysis, can be assessed against.

2.3.2 What criteria are tools evaluated against?

The criterion against which digital forensic practice is measured is based on best practice guidelines and the international standards that have been produced. A review of these follows below.

Computer Forensic Tool Testing (CFTT)

The Computer Forensic Tool Testing (CFTT) project at the National Institute for Science and Technology (NIST, 2003b) has undertaken a number of tests against carefully crafted specifications authored by them. The testing methodology used is developed by a steering committee of law enforcement & NIST staff (NIST, 2003a). Considered by some to be rigorous (Liang, 2010), the project has been credited with identifying at least one issue that might otherwise have been undisclosed by the vendor concerning the last sector on hard disk with an odd number of sectors that was not acquired using the tool *dd*. Although subsequently found to be a Kernel and not a software tool issue (Flandrin, Buchanan, Macfarlane, Ramsay & Smales, 2014), it highlights both the benefits of a thorough testing regime and the risk of misinterpreting the results.

Validity

One of the biggest drawbacks of the CFTT project concerns its validity. Critics of the project, argue that it is largely focused on acquisition (Guo & Slay, 2010a), (Newsham, Palmer, Stamos & Burns, 2007) and Sommer (2010) who points out the tests completed by CFTT are but a "tiny subset" of the functionality that needs to be tested. Dykstra & Sherman (2012) agree, adding that in a climate of ever increasing cloud based forensics none of the enterprise versions of products (that include remote forensic capabilities) have been tested. Furthermore, it is not just the tested functionality of the CFTT project that is considered to be too narrow; Guo & Slay (2010c) point to inadequately sized test spaces, where there is a need for large reference sets to cater for possibly "thousands" of possible scenarios to validate just a single function.

Beyond concerns over the scope of what is tested, there are also challenges of validity surrounding the timeliness of the test specifications and the tests themselves. For example, the data acquisition specification is in draft and is over ten years old (NIST, 2005). The document states it's scope is to cover "ATA, SCSI, USB, or Firewire interfaces". Therefore, no provision is made for more recent technology, such as Solid State Disks. Furthermore, Flandrin et al. (2014) point out that many of the tests performed take too long to publish and so are on older, depreciated versions of software.

Commercial vendors of products tested by the CFTT project seek to minimise the impact of this issue. In 2011 Guidance Software Inc. (GSi), who produce of the forensic imaging/analysis application named "EnCase", stated that the CFTT project demonstrates, "rigorous and comprehensive testing" of EnCase 3.20 (Guidance Software Inc., 2011). The test results date from 2003 and the current release is of this product is v7.12. GSi point out that "no substantial changes" to the imaging functionality of the product has taken place since v3.20 of the product. Clearly, there is a potential challenge of bias here that contravenes the scientific method (see Table 1-1). In addition, critics would argue that the meaning of 'substantial' is unclear here. Furthermore, the software for this functionality may have changed little over time, but the hardware it is interacting with has changed significantly.

Field maturity

The fledging nature of the digital forensics field and its rapid evolution has led to an ad-hoc development of the CFTT project. According to Beckett (2010), the field has not been mapped out sufficiently prior to undertaking the project.

Test results

The overall aim of the CFTT project is to provide feedback to "improve tools" and for users to make "informed choices" in selecting a tool (NIST, 2003b). However, it could be argued that the problem with assertion based tool evaluation (as advocated by the CFTT project) is that the outcome is either a pass or fail. Such tests do not inform the reader if it was a bare pass or a substantive one (Peisert, Bishop & Marzullo, 2008). Saleem *et al.* (2012) make the same observation and adds that no "comparative study is conducted to help an investigator in selecting a better tool". Furthermore, Byers & Shahmehri (2009) point out that results are reported without any deeper analysis as to why a given test has failed.

Scientific Working Group on Digital Evidence (SWGDE)

Unlike NIST who developed specifications, plans and assertions, SWGDE have developed a more relaxed approach to forensic tool testing by producing test guidelines and templates (Liang, 2010).

However, a significant problem with this approach is that their test results are only available to US law enforcement agencies. Flandrin *et al.* (2014) point out that this decision is contrary to the principle tenet of information sharing in science. Hence, it could be argued that this lack of reproducibility (see Table 1-1) has rendered any results obtained from such tests to be non-scientific. Ironically, the decision also runs contrary to SWGDE's own advice, advocated in one of the few documents they have released publically, where they call for test results to be repeatable (SWGDE, 2012).

Aside from the matter of reproducibility, the reasoning demonstrated by the group may be subject to challenge. SWGDE (2008) have openly stated that in computer forensics "false positives are non-existent". However, the forensic product EnCase has previously been subject to a software bug that resulted in data in the first 4GB of unallocated clusters being duplicated (creating false positives) through to the last unallocated cluster (Sanderson, 2008). In another example, the Linux version of the EnCase imaging tool (named *LinEn*) was found to "insert sectors into the image that were not present on the drive" (Byers & Shahmehri, 2009).

Department of Defence Cyber Crime Center (DC3)

As with the SWGDE, the DC3 circulate the results of their tool testing only to a closed group of individuals and not the wider scientific community, thereby once again rendering the scientific validity of such tests open to challenge on the grounds of reproducibility, see Table 1-1. A list of tools reportedly tested is published, which include commercial forensic tools such as EnCase. The list of reported versions for this tool includes 7.09.02, 7.08, 7.06, 7.05.02.03, 6.19.7, 6.18.0.59, 6.15.0.82, 6.13.0.43, 6.11 (DC3, 2016). This is more comprehensive and up to date than the list published by CFTT. Without access to the results or even the testing methodology employed by DC3, the approach is of little value to this research project.

Forensic Science Regulator (FSR) Codes of Practice and Conduct

As indicated in section 2.1, the United Kingdom is moving to adopt the BS EN ISO/IEC 17025:2005 Standard (ISO, 2005), referred to hereafter as "the Standard". This Standard is incorporated into the Forensic Science Regulator's Codes of Practice and Conduct (2016).

Traceability of reference data sets (FSR: 22 | Standard: 5.6)

Becket (2010) refers to section 5.6.3.2 of the Standard, stating there is a "need" for forensic practitioners to demonstrate that "certified reference materials" have been used to evaluate their tools. This is not quite accurate as the same section of the Standard states this should be done "where possible". A small number of attempts over the years have been made by the scientific community to address this lack of standardised test data. However, none of the following datasets have been labeled *certified*.

To bridge the gap between the tests produced by the CFTT project and the needs of practitioners, Carrier developed a series of Digital Forensic Tool Testing (DFTT) images (2010). The datasets are quite old and mostly date from 2003-2005, with one entry for 2006 and a final entry in 2010. The approach taken by Carrier is to fabricate the data with documented features. A limitation of this is that variations that arise in normal operation are not present in the data (Casey, 2011b).

A more comprehensive series of test images is provided by NIST for their Computer Forensic Reference Data Sets (CFReDS) project (NIST, 2016). Different groups have developed the datasets and the supporting documentation is incomplete in some cases (Casey, 2011b).

It is argued by others that both the DFTT and the CFReDS projects may be useful for teaching but less so for tool testing as not all functionality can be tested and their static nature means they cannot be extended (Flandrin, Buchanan, Macfarlane, Ramsay & Smales, 2014).

Garfinkel *et al*. (2009) developed an extensive collection of both fabricated and real data, captured from physical devices purchased second-hand from around the world. This material is intended to be used for "computer forensics education research" (Digital Corpora, 2017) and so is not intended for tool testing. As before, Casey warns that the supporting documentation is incomplete in some cases.

Further to the cited limitations above, none of the datasets above are specifically known to contain malware. Hence they have not been included in this research.

Estimate of uncertainty (FSR: 20.18 | Standard:5.4.6)

In addition to the traceability of reference material, section 20.18 of the Standard states that a review of the uncertainty of measurement shall be made when procedures are modified (or initiated). Also, section 25.2.1 (c) states that practitioners who provide reports to the CJS should be able to demonstrate the impact that a given measurement uncertainty has on a given conclusion.

Furthermore, Beckett argues that both the SWGDE and NIST test methodologies "ignore" this "critical" element of ISO17025. He argues that both groups do not provide an estimation of uncertainty of measurement.

Validation of software (FSR: 20.2 | Standard:5.4.5)

Gallop and Brown (2014) argue that even if forensic labs achieve ISO 17025 accreditation as a minimum standard, it is insufficient to service the needs of the CJS. They further argue that the FSR is taking a "light touch" to the matter of accreditation. Qualifying terms such as, "where possible" (see above) perhaps evidence this. They conclude that the FSR Regulator "may not be sufficiently stringent" to sufficiently quality assure all forensic science activity.

Marshall (2010) takes the view that to comply with ISO 17025, an organisation must be in a position to demonstrate that their tools, procedures and methods are fit for purpose. To achieve that, he goes on, validation and verification would need to be applied. Furthermore, he argues that validation and verification need clear requirements, which are not properly documented anywhere.

Marshall is also the editor for the more recent ISO 27041 standard (ISO, 2015) which proposes using verification, validation and acceptance for evaluating digital forensic software. The standard seeks to overcome the problem of digital forensic software developers not releasing (or even producing in the first place) formal requirements specifications that would facilitate validation testing (Casey, 2012). The standard places the onus on the developers of forensic software to provide evidence that their tools meet the prerequisite requirements set by accredited digital forensic laboratories. Whoever sets the requirements, Flandrin *et al.* (2014) warns that the evolving nature of the field is such that the time to define the requirement for a single function "need to be counted in years".

With so many issues surrounding the criteria against which forensic software tools would be validated, it is worth taking a moment to consider the benefits that stand to be gained as a result of such a process.

2.3.3 Benefits of evaluation

As discussed in section 1.1.2, courts are moving away from a default position of trusting expert evidence. The practice of naïvely accepting anecdotal assertions from experts on tool reliability is effectively discouraged by the FSR. Furthermore, evaluating forensic tools should help to minimise flawed technical evidence, as identified in section 1.1.3, providing a mechanism for a sitting judge to assess reliability of expert evidence.

Providing a more scientific footing for the evaluation of forensic tools, especially if they are developed over time, could give evidence of a tool's reliability. If such evidence were shared amongst the community, this would contribute to a body of knowledge for that tool. Moreover, as discussed in section 1.1.5, a clear methodology and record of test results would also facilitate the repeatability of a tool's behaviour under a given set of circumstances. This would assist in enhancing the scientific credibility of the tool from the CPS's perspective, see Table 1-1.

Whilst some testing has been documented for existing forensics tools, nothing has been identified for tools used for investigations involving malware. Such tools will be required to meet the statutory requirements just as much as conventional forensic tools. A mechanism to evaluate such tools in accordance with the ISO 17025 standard would contribute to addressing this gap.

It has been argued that international standards, such as ISO 17025 which underpin the statutory requirements set by the FSR, promote market efficiency and expansion, foster trade, encourage competition and lower barriers to market entry (Guttman, 2009). These commercial benefits are perhaps less applicable within the Criminal Justice System, but are more relevant between forensic service providers competing for contracts with law enforcement agencies. What is perhaps more important for digital forensic practice as a whole, is the need to minimise miscarriages of justice resulting from poor working practices.

Implementing a tool evaluation strategy is not without its risks and challenges. The next two sections will consider these in brief.

2.3.4 Risks to tool evaluation

The Oxford English Dictionary (2016b) defines the term risk as being the exposure to "the possibility of loss, injury, or other adverse or unwelcome circumstance; a chance or situation involving such a possibility". It could be argued here that the loss incurred amounts to anything that undermines the credibility of the results. Perhaps the most significant of these is the risk of misinterpreting the true cause of an identified error in a tool's output.

Flandrin *et al.* (2014) cites a test report produced by NIST (2002) that indicated that the data acquisition tool *dd* had been unable to acquire the last sector from a disk containing an odd number of sectors. It transpired subsequently that the cause of this anomaly was not a fault with the tool, but with the kernel of the Linux operating system where the test was performed. Mitigating against this type of risk is not easy. For example, consider the possibility of

apparently perfect test results arising from a fault in a tool running on such a kernel, whereby the last sector acquired was a duplicate of an earlier sector. The additional sector erroneously captured by the tool would cancel out the effect of the missing sector dropped by the kernel bug.

Even if such a risk is mitigated, it is imperative that the individual conducting the test has specialist skills to ensure the tests are conducted in a scientifically valid and repeatable fashion to ensure consistency (Pan & Batten, 2009). Lyle (2010) argues that many of the procedures followed by practitioners contain errors made that are systematic, rather than statistical in nature. Furthermore, this argument is readily extended to include the reporting on the results of the test, which would typically require sufficient statistical skills.

Deliberate attempts to invalidate the results obtained from tools are an objective for anti-forensics. Anti-forensics is the use of techniques to invalidate the findings of a forensic investigation. Hence anti-forensics techniques are a risk to the validity of tool evaluation. Shanmugam (2011) considers the impact of such techniques and using a combination of the CFTT and DFTT frameworks, he develops a technique to apply what he terms "meta-forensics" to recognise and thus counter anti-forensic techniques.

Even if all of the above risks were mitigated, there remain a number of challenges to be faced for the evaluation of forensic tools, particularly when applied to a malware investigation context.

2.3.5 Challenges of tool evaluation

Bias

The Forensic Regulator's Codes of Practice and Conduct (Forensic Science Regulator, 2016) incorporate the principles of the ISO 17025 Standard (ISO, 2005). Section 5.4.5.3 of the Standard states that the range and accuracy of values "shall be relevant to the customers' needs". Hence an element of systematic bias is introduced into the implementation of the Standard, thereby opposing one of the attributes of the scientific method (see Table 1-1). This drawback is recognised in the Standard as a "balance between costs, risks and technical possibilities". Section 5.4.5 of the standard outlines the requirement for the testing laboratory to perform validation on "non-standard methods, laboratory-designed/developed methods, standard methods used outside their intended scope, and amplifications and modifications of standard methods". The pace of change of the technology surrounding digital data is such that this requirement would apply to almost any forensic investigation performed, as tools that have yet to be updated are applied to more recent (and untested) forms of the data under analysis.

Pace of change

Although this pace of change is high, it is perhaps not as extreme as the rate at which malware evolves (Rieck, 2008), (Ashford, 2010). One report (G Data Software AG, 2016) suggests that on "average" a new malware sample is identified at the rate of approximately one every six seconds. Although the report does not make clear what is meant by "average", it can be argued that few if any applications and technologies (such a new social media platforms) that need to be analysed for forensic artefacts evolve at such a rate. It is not unreasonable to suggest that the tools used to analyse such malware could become deprecated equally as quickly.

It is not just the tool's capability that may be wanting; with a constantly developing field, another challenge faced by the profession is that the testing of such tools typically lags behind the current release of a given tool (Flandrin, Buchanan, Macfarlane, Ramsay & Smales, 2014). Part of the reason for this maybe the length of time it take to formally publish results from such tests (Sommer, 2011). Another reason may be the sporadic nature of the field's evolution.

Ad-hoc evolution

Some consider that the digital forensics field advances in a reactive and not a proactive manner and that it is conducted not to develop the field but to "quell criticism over a technique's accuracy" (Cooley, 2004). Others who suggest that it is crime that drives the field and not scientific enquiry echo this viewpoint. Hence, they argue, digital forensics "follows the trend rather than leading it" (Raghavan, 2012).

Reproducibility

A tenet of a scientific method is that it is reproducible (see Table 1-1). Wilson & Slay (2006) argue that the use of reference sets is "critical" to effectively evaluate a tool's "correctness". Garfinkel *et al.* (2009) agree and point out that without reference data sets such reproducibility is not possible, as others cannot validate the techniques developed and tested. In terms of sourcing the data for such data sets, they go on to warn of the problems of using real data, citing issues of privacy, copyright and other legally protected material. To circumvent this issue, Garfinkel *et al.* offer both real and fabricated data for the teaching, research and the evaluation of tools.

Closed source issues

However, Beckett & Slay (2007) argue the validation of the discipline is non-trivial and requires a structured framework that the ISO 17025 Standard does not address. To illustrate this point, they highlighted the use of closed-source tools requires a subjective judgment on the part of the individual undertaking the test to produce a test plan that is sufficient on both coverage and depth to identify any validation issues. Casey (2012) argues that this results in practitioners and tool testers making "educated guesses about how a given tool works". It is perhaps for these reasons, argue Beckett & Slay, that the definitions within the Standard describe only the outcome and not the tools or methodology taken to achieve it.

2.3.6 Who does the evaluating?

The question of who performs the evaluation of software for use in a digital forensic environment is addressed by the FSR in their Codes of Practice and Conduct in section 20.2.1 which states that the forensic laboratory (provider), vendor or another provider may perform the validation:

> *Validation should be conducted prior to implementation of the method. This may be performed by the provider, manufacturer or another provider.*

What follows is a brief review of each of these groups.

The software vendor

The closed-source nature of commercial forensic software may be one factor that has led to practitioners relying too much on software vendors testing their own software (Flandrin, Buchanan, Macfarlane, Ramsay & Smales, 2014). However, this practice does not ensure the practitioner is compliant with the ISO 17025 standard, as the local environment under which the software and any equipment is used can impact on the results (Beckett & Slay, 2007).

An interesting response to this is the emerging ISO/IEC 27041 (ISO, 2015) standard (discussed in section 2.3.2 above) which sees the forensic laboratories setting the requirements that software vendors must provide evidence of satisfying through testing. In principle this makes sense, however a vendor would not be able to test any given software tool in every conceivable environment the product may be used in.

In addition, section 21.1.3 of the FSR's Codes of Practice and Conduct states that "User acceptance testing shall be performed prior to software and/or related equipment being placed

in service". Hence, compliance with the Codes of Practice and Conduct would still require a minimum level of testing to be performed by the Practitioner/Forensic laboratory.

Furthermore, although multiple vendors would be required to meet the same standard under this scheme, the design, methodology and conditions of any testing they perform would very likely be different from one vendor to another. There is therefore a risk that two similar products have not been tested under the same conditions. Hence this poses a threat to the scientific validity in terms of reproducibility, see Table 1-1.

Clearly, as a software developer, it would be unreasonable to expect a vendor to produce and ship code without any form of testing. Hence, argues Dow (2007), the practitioner would be dependent on the vendor to a degree to undertake some form of testing. The exact level of testing, he continues, would be subject to a level of cost needed to keep the tool affordable, thereby imposing practical limits in testing that can be done. Dow concludes with a warning that a vendor testing their own product is subject to a conflict of interest and would be likely to be reticent to reveal problems. Hence, the scientific validity in terms of bias could be impacted by this approach, see Table 1-1. For these reasons, a more independent body would be a preferred solution.

Independent body

One approach to overcome the problem of inconsistent test conditions, is for the testing to be centralized and made accountable to one or a small number of independent testing bodies, such as CFTT, Underwriters Laboratory (2016) or the Common Criteria (2016). The CFTT project was instigated with this purpose in mind, but as stated in section 2.3.2, is subject to a number of challenges, rendering it not viable for law enforcement agencies. Dow (2007) points out that the funding for testing by such organisations is unclear and hence the viability of their ongoing testing commitment is uncertain. Furthermore, Dow argues that until an official and funded resource is available practitioners have no choice but to do testing themselves. However, practitioner based testing also poses a number of challenges as well.

Practitioner

When operating within the criminal justice system, it is the practitioner who tenders evidence and is therefore ultimately accountable for the reliability of such evidence. Hence good practice dictates that as a practitioner you would test a new (or an established, yet unfamiliar) software tool on a known dataset to be satisfied that your conclusions are sound. Such testing should be "regression testing" (Beckett, 2010) to account for any bug fixes or enhancements

made to the software. This requires significant resource on the part of the practitioner. One way to alleviate this pressure might be to centralise the test results within a team or organisation. However, given practitioners are accountable for their own work (Fab4, 2011) and following a Supreme Court Judgment in the USA where they are now subject to being sued for professional negligence (Supreme Court, 2011), it is unlikely that many practitioners would feel comfortable relying on the work of others to underpin their evidence.

Nevertheless, practitioners are busy people with heavy caseloads and the time for developing and executing extensive tests on tools would be a significant challenge for most of them (Dow, 2007). Flandrin *et al.* (2014) agrees, adding that most practitioners have a limited number of resources. As a result, they are not in a position to "test all tools along with all versions".

2.4 Chapter summary

This chapter has briefly outlined current digital forensic practice and the elements of the regulatory requirements salient to this research. The lack of publications on the impact of applying this to a malware investigation has also been highlighted, noting in particular the lack of a viable method for evaluating tools used in a forensic investigation involving malware. A critique of the tools and techniques available to study malware as part of an investigation was explored and concluded with a discussion on how such tools can be evaluated to meet the criteria laid down by the Forensic Regulator.

Evaluating tools against criteria set by the Forensic Science Regulator has several benefits. To start with, the approach proposed by this research would benefit from an established credibility, as it would potentially meet both the Regulator's Codes of Practice and the underlying requirements of the ISO 17025 standard. In addition, conformance to established evaluation criteria would arguably make the approach more familiar and easier to adopt into working practice. Finally, alignment with the ISO 17025 standard would potentially make the approach scalable, as the validation process of the incoming ISO 27041 standard is "compatible" with ISO 17025 standard's validation process (Marshall, 2011).

Alongside the benefits this chapter also considered the risks in section 2.3.4. The complexity of these risks means that not all of these identified risks will be addressed by this research. Alongside the benefits of speed, the use of automation would help to minimise the risks associated with a practitioner's lack of skills in the fields of statistics and scientific research. The interpretation of results from a tool can also, of course, be impacted by the presence of any anti-forensic measures present in the malware. Full mitigation against these measures is

complex and outside the scope of this research. However, large scale testing and statistical reporting again offer a means to identify errors in the data, resulting from causes such as this.

The challenges to evaluating tools were explored in section 2.3.5 and explored issues of bias, pace of change, the ad-hoc evolution of the field, reproducibility issues and the use of closed source tools. To address the issue of bias in practitioners who may strive for a required level of accuracy, this research will report its findings with a stated level of statistical confidence and leave the rounding process for the consumer of the report.

To counter the challenge concerning the pace of change, it is important the approach offered by this research has a relatively short test time. Hence, by providing the practitioner with an automated solution to evaluate a tool against a large bank of malware in a relatively short space of time the impact of changes in the technology can be minimised. To address the reproducibility concerns, it is proposed that the Malware Analysis Tool Evaluation Framework (MATEF) together with the test data of binary malware files (as implemented, discussed later in section 4.3.2) be made available to the academic community.

This chapter has identified varying criteria used to evaluate tools with particular focus on the FSR's Codes of Practice and Conduct. The following chapter synthesises malware forensic practice with the FSR requirements, legal requirements and technical recommendations to develop a single set of requirements for tools used in a malware forensics environment.

Chapter 3 Malware tool evaluation requirements

The previous chapter included a review of existing malware forensic practice and determined that there is little published research into the area. Hence there is little support for any trust placed in such practice, see the first research goal of Table 1-2. The chapter went on to identify the five-phase malware analysis model of Malin *et al.* (2008). In this discussion, it was argued that despite the ad-hoc nature of the model, this was the most viable starting point for this research. Furthermore, the fifth phase of this model (dynamic malware analysis) was selected as the basis for this research, as the use and analysis of the tools within this phase was deemed the most achievable within the constraints of the research.

Given the lack of a scientific methodology to perform malware forensics, this chapter draws its attention to identifying the requirements of such a methodology in order to subsequently design a solution (see second research goal, Table 1-2). The approach taken is to start with identifying the themes that are apparent from the research question. Hence the chapter opens with a section (3.1) that explores these themes before moving on in the next section (3.2) to determine the existing requirements; thereby providing a context. These themes and requirements are then synthesised in the next section (3.3) to formulate a set of proposed requirements, designed to address both the research question and the existing requirements. The chapter closes with a discussion (see section 0) on the analysis and design methodology chosen.

3.1 Interpretation of the Research Question

To recap, the Research Question in section 1.2 stated:

> *Can a systematic basis for trusted practice be established for evaluating malware artefact detection tools used within a forensic investigation?*

Three broad themes were apparent from this question, namely trusted practice, tool evaluation and forensic investigation.

Trusted practice

The first of these, trusted practice stems in part from the unfounded trust placed in tools. The review of the current state of the field highlighted that a largely non-scientific and anecdotal approach is adopted by some practitioners who either rely upon repeated confirmation to establish truth and/or accept the results of digital forensic tools solely on the reputation of the

vendor (section 1.1.2). Furthermore, this lack of trust is compounded further by problems with expert evidence and that the practice of withholding test results from the scientific community by groups such as SWGDE and DC3 do little to instil confidence in trusted practice (section 1.1.3).

Turning from what a lack of trust looks like to how it is defined in this research, recall from section 1.2 that the definition of trusted practice applied in this research is derived from the Crown Prosecution Service (CPS) (2015), who state that expert evidence must be reliable and hence have a "scientific basis". As a result, five attributes of the scientific method were identified, i.e.: Repeatability, Reproducibility, Testable hypothesis, Controllable and Unbiased, see Table 1-1. For a malware analysis tool to be evaluated in a manner that addresses the Research Question, the extent to which the evaluation methodology meets these five attributes of the scientific method should be assessed.

Malware tool evaluation

The second theme apparent from the research question was that of malware tool evaluation. One of the contributions of this research is to address the lack of material published on evaluating tools used to analyse malware (see section 2.2). Specifically, there is currently no definition or criterion to describe tool evaluation in a malware context (see section 2.3.1).

An important element of evaluation is to identify what exactly is to be evaluated. Therefore, the sections that follow identify and develop the requirements and consider how best to meet them. Furthermore, consideration has been given as to how this evaluation is reported. The assertive pass/fail reporting of the CFTT (see section 2.3.2) lacks the granularity to distinguish between tools that pass a test with a narrow or comfortable margin. Hence, it is not possible for the practitioner to choose the better of two tools evaluated in this way.

To evaluate every aspect of malware analysis tools was outside the scope of this research. To keep things focused, consideration was only given to tools that identify malware artefacts that hence assist in the understanding of malware behaviour (see section 2.2) as part of a forensic investigation. No documentation has been found to map out the functionality of tools used for investigating malware for this purpose. Furthermore, no framework has been identified to systematically test such tools.

Forensic investigation

The final theme identified from the research question was forensic investigation. Therefore, the trusted practice identified above relates to work undertaken within the criminal justice system, which carries with it various implications. For example, the processes applied to

undertake investigations are subject to legal requirements. For instance, steps should be taken during an investigation to ensure that malware is not permitted to gain unauthorised access to resources or to exfiltrate personal data. Furthermore, the output of such an investigation is subject to legal admissibility requirements. A lack of scientific principles and provenance in expert evidence could lead to expert evidence being deemed inadmissible (Law Commission, 2011).

Therefore, another implication for operating within the criminal justice system is the growing need to operate within regulatory requirements (see section 1.1.7). The current regulatory requirements are the Forensic Science Regulator's Codes of Practice and Conduct (2016). Not all police forces are committed to meeting the required standards, leading the Forensic Science Regulator to warn that the "integrity of the criminal justice system in England and Wales is under threat due to the quality of forensic science work" (Toner, 2017).

3.2 Existing requirements

The legal and regulatory implications outlined in the previous section can be managed by identifying the requirements to operate both lawfully and in a manner that maintains a minimum standard of quality. A minimum level of quality would in turn help to instil a greater level of trust in the evidence produced. To implement these requirements, controls in the form of technical measures are needed. Hence the remainder of this section is divided into three sub-sections to explore the technical, legislative and regulatory requirements associated with the forensic analysis of malware behaviour.

3.2.1 Technical recommendations

A review of the literature determined that little published or otherwise formal requirements for a technically valid malware analysis lab have been proposed (see section 2.2). The closest there is to such a requirement is the phased series of guidelines offered by Malin *et al.* (2008), presented in section 2.2. However, Malin *et al.* recommend "flexibility and adjustment of the methodology" to cater for the needs of each case under investigation. Hence it is difficult to stipulate that a specific series of processes should be followed to perform malware analysis. Nevertheless, several recommendations that could be applicable to almost any malware forensics investigation were identified. The first of these is the use of virtual machines.

Virtualisation

The use of virtual machines (VMs) is recommended by Ligh *et al.* (2010) who also stipulate that such software should be updated frequently to minimise the risk of exploits being used to enable the malware to break out of the virtual environment onto the host. They also advise

that shared folders on the host be either disabled or read-only. They further suggest that access to resources such as a network or removable media be disabled. Sikorski & Honig (2012) agree suggesting that VMs should be configured to be a 'host only' network, meaning the virtual network on which they reside should be isolated from the physical network on which the host resides. Szor (2005) points out the need to reset a test system to a clean state and hence promotes the VMs for their speed at resetting. The use of VMWare (VMWare, 2016) is cited by Szor as a good choice for this, though little mention is made of any other virtualisation solutions other than a passing mention of Microsoft's Virtual PC.

Binu and Kumar (2011) evaluate two alternative virtualisation solutions, based on the hypervisors KVM (https://www.linux-kvm.org/) and Xen (http://www.xenproject.org), concluding Xen to be superior in terms of performance and stability.

Network service provision

Isolating malware from a network or even the Internet could limit the behaviour exhibited. To counter this, it is a good idea to provide the malware with as many services as possible that it is likely to rely upon, such as SMTP, HTTP and DNS. Wagener, Dulaunoy & Engel (2008) reply upon trapping DNS queries from malware using a local DNS server. Sikorski & Honig (2012) suggest the use of INetSim (Hungenberg & Eckert, 2016) to simulate a broader range of network services. Palkmets *et al.* (2014) also deploy INetSim but additionally provide a route to the Internet via an onion router network.

Although the exact services needed would be dictated by the malware that is executed, a simpler requirement would be to provide as many services as possible.

Resource Monitoring

Given the provision of network services highlighted above, Malin *et al.* (2012) advise that network monitoring be put in place to observe any attempts by the malware to resolve DNS queries or to connect to remote IP addresses. They also advise monitoring the access made to processes, files, API and the Windows Registry. As a starting point to identify monitoring tools, Liao & Langweg (2014) review a number of systems for both Windows and Linux environments, designed to perform monitoring in a variety of ways.

Szor (2005) also recommends monitoring file, registry, network, system calls and process monitoring, but warns that only a combination of monitoring and detailed disassembly will reveal the entire functionality of malware. This warning is not applicable to this work, as the scope of the research does not include malware analysis; what is in scope is the evaluation of the tools used to do such analysis, see section 1.2.

Vulnerable environments

Similar to the provision of a networked environment, Szor (2005) also argues that many malware threats are vulnerability dependent and so failure to provide a suitable fertile environment could lead to a failure in the malware activating. To this end, Szor advises that unpatched, older versions of software be used. Arguably, this could be extended to include recent but not current versions of operating systems as well.

Malware handling procedures

Malware is like a hazardous substance and needs careful handling to avoid unwanted contamination of an organisation's production/corporate network. Szor (2005) warns that some analysis tools can result in the unexpected execution of malware as part of the analysis. Tools such as *PEiD* (Aldeid.com, 2017) which detect packers used to obfuscate malware and *IDA Pro* (Hex-Rays, 2015) used to disassemble/debug binary code both execute the binary under analysis as part of their normal operation. Szor adds that the source of some tools also means that either the website they are obtained from or even the tool itself can be laden with malware.

3.2.2 Legal Requirements

The primary focus of this research is UK practice, hence the requirements directly applicable to this jurisdiction are considered over and above those of other jurisdictions. The legal requirements surrounding tools used to evaluate malware can be divided into two broad areas. The first is the legislation concerning the risks associated with handling the tool's test data (ie: malware). The second concerns the admissibility requirements of the output produced by the tool under evaluation. If submitted as evidence to the Criminal Justice System, the tool's output must adhere to these strict criteria.

Handling malware

To evaluate malware analysis tools the test data used should ideally be real malware. Further to the technical recommendations identified in section 3.2.1, the handling of malware is also subject to legal restrictions that impose tight controls on the handling of such malware.

Without appropriate precautions to limit the reach of the malware, execution of such malware could result in unauthorised access to a system, thereby breaching the Computer Misuse Act (1990). Furthermore, such malware may also scan the local network and harvest personal data with a view to the exfiltration of this data to a third party. Such behaviour may land the data controller of the victim network liable under the Data Protection Act (1998). Even with

these controls in place, there are also requirements in place for the material produced from a software tool to qualify it as admissible evidence.

Admissibility

For the output of a malware analysis tool to be to be tendered as evidence, the output itself needs to be admissible. As mentioned in section 2.3.1 digital evidence is presumed to be valid and thus may be admitted unless evidence to the contrary is provided (Lloyd, 2014). Superficially, this may seem to suggest that there is no need to prove the validity of the data produced by a software tool used to analyse malware.

However, Lloyd goes on to argue that the general precept on the 'hearsay' rule is that evidence must relate to actual knowledge rather than what has been told to a witness (which, argues Lloyd, can be a human or machine). Hence any data produced by a computer could be deemed hearsay and (in line with section 129 of the Criminal Justice Act 2003) can only be admissible if proven to be accurate by a suitably qualified expert.

Lloyd further argues that because of R v Shepard [1993] AC 380, this heavy standard of proof is reduced when a person familiar with the expected output of a computer is available to give evidence. However, it could be argued few persons would be familiar with the expected output of a tool used to analyse malware, which typically produces random artefacts. Hence, such tools perhaps should not be used without relevant expert testimony. This makes it important to test these tools in a robust way that can demonstrate their reliability so that the expert testimony is more credible.

Reliability

Guidance on expert evidence from the Crown Prosecution Service (CPS) (2015) states that expert evidence will be admissible under common law where:

- It will be of assistance to the court
- The expert has relevant expertise
- The expert is impartial
- The expert evidence is reliable

The first of these requirements concerns the forming of a judgement on the relevance of the evidence tendered, whilst the second and third concern a judgement on the expert. The last requirement concerns both the evidence and the manner in which it was produced.

Recall from the section 3.1 that in this guidance the CPS state that reliable expert evidence must have a "scientific basis" and that five attributes of the scientific method were identified,

i.e.: Repeatability, Reproducibility, Testable hypothesis, Controllable and Unbiased, see Table 1-1. Therefore, for a malware analysis tool to be evaluated in a manner that addresses the theme of *trusted practice* in the Research Question, the extent to which the evaluation methodology meets these five attributes of the scientific method should be assessed.

The fast evolving nature of the IT field make it particularly susceptible to challenges on the reliability of evidence produced using fledgling techniques. Despite these concerns, in R v Clarke (RL) [1995] 2 Cr. App. R. 425 Lord Justice Steyn concluded that it would be "entirely wrong to deny to the law of evidence the advantages to be gained from new techniques and advances in science". Subsequent to this ruling, the CPS produced guidance on the use of novel evidence that is based on the judgement of R v Lundy ([2013] UKPC 28) and is set out in Table 3-1:

#	**Guideline**
1	Whether the theory or technique can be or has been tested
2	Whether the theory or technique has been subject to peer review and publication
3	The known or potential rate of error or the existence of standards
4	Whether the theory or technique used has been generally accepted

Table 3-1 : R v Lundy Guidelines

These guidelines have been woven into regulatory requirements that are slowly becoming mandatory for forensic practitioners who wish to submit evidence to the Criminal Justice System in the UK.

3.2.3 Regulatory Requirements

Regulation is still within its infancy within the UK, hence the Codes of Practice and Conduct (2016) of the Forensic Science Regulator have yet to be fully implemented. The Codes state that "irrespective of whether the provider is public, police or commercial" all digital forensic providers will be required to demonstrate they are accredited to ISO/IEC 17025 and the Codes of Practice by October 2017. This deadline applies to imaging, data recovery using Commercial Off The Shelf (COTS) products, extraction and analysis of data.

Some of the regulatory requirements are linked to legal guidance. As previously stated, the Codes contain the guidelines set out in R v Lundy ([2013] UKPC 28). Guideline 1 (see Table 3-1) can be linked to section 20.1.5 of the Codes which states that for novel techniques the provider "should have validated the method, product or service". The second guideline from Table 3-1 concerns peer review which is addressed by section 20.16.1 of the Codes. Section 25.2.3(e) of the Codes address the fourth guideline on the level of peer acceptance for a

technique. However, the third guideline concerning the rate of error is not addressed either by the Codes, the associated draft guidance tailored to the validation of digital forensic methods (Forensic Science Regulator, 2015) or the underlying ISO/IEC 17025 standard. Part of the reason for this may be a lack of understanding of the term, a lack of sufficient training in statistics and the scientific method, or even the concern that "current methods will be exposed as lacking an empirical basis" (Christensen, Crowder, Ousley & Houck, 2014).

Furthermore, it is worth noting that the Forensic Science Regulator warns against validating only a tool rather than the method it is part of in section 13.3 of their Consultation document (2015). However, as reported in 2.3.1, what little that has been published to date focuses on methodologies and not tools. This leaves a gap in the validation process, which forms the basis of this research. Hence, the focus of this research is to provide a framework to evaluate the tools as part of a wider method evaluation.

The following section examines technical, legal and regulatory requirements oulined in this and the previous two sections to synthesise a set of proposed requirements.

3.3 Proposed requirements

Recall from Chapter 2 that studies concerning the impact of regulatory requirements on malware forensic practice are lacking. In particular, there is a clear need for a methodology to evaluate tools used in digital forensic investigations involving malware (see section 2.4). This section will explore strategies for satisfying the Research Question in light of the existing requirements identified in section 3.2.

The Research Question (see section 1.2) requires that a level of trusted practice be established. Fundamentally, trust can be considered to involve "willingly acting without the full knowledge needed to act" (Duranti & Rogers, 2012). In the context of the Criminal Justice System involving expert evidence, this arguably translates to a Court coming to a decision on the reliability of a given piece of such evidence based upon two forms of trust.

The first of these is the trust in the interpretation or impact of the evidence provided to the court. This trust is placed upon the expert presenting the evidence. To assist the court in its deliberations, the expert provides an interpretation on the meaning and impact of the evidence tendered. The outcome of such deliberations ultimately considers the bearing such evidence has on the case as a whole. Given this and that such trust is based upon the expert's knowledge and skills as well as their ability to communicate these effectively, this form of trust is outside the scope of this research.

The second form of trust is that placed on the reliability of the evidence tendered. Since the repeal of section 69 of the Police and Criminal Evidence Act 1984, any evidence produced by a computer is presumed to be reliable; hence it is therefore admissible (CPS, 2014), see also section 3.2.2. However, the motivations for this research identified in section 1.1.1 indicate this trust has been undermined. The Forensic Regulator's Codes of Practice and Conduct (2016) provide an independent vehicle to instill a level of assurance in such trust. Hence as a requirement, trusted practice has moved from inherent and internal to external in nature.

Forensic investigation also forms an element of the Research Question and therefore, given the motivation for the research sits within the Criminal Justice System (see sections 1.1.1 and 1.1.3), is subject to externally set admissibility requirements (see section 3.2.2) and regulatory requirements (see sections 1.1.7 and 3.2.3).

The Research Question also requires the evaluation of tools for malware artefact detection. However, this is not currently subject to externally set criteria. For example, both sections 20.2 (validation of methods) and 20.8 (validation of measurement-based methods) of the Codes of Practice and Conduct (2016) make no requirement for accuracy. The latter of these

two sections does state the results must be "consistent, reliable, accurate, robust and with an uncertainty measurement" but this does not specify the level of accuracy required.

This differs to fields such as engineering where the specification drafted by the client might require a component to have a property that is within a tolerance of a given specified value. Hence, the evaluation of tools for malware artefact detection is an internally set requirement.

From the body of existing requirements in section 3.2, candidate requirements were considered for inclusion in the proposed requirements list based on the methodology illustrated in Figure 3-1. This procedure is analogous to the 'Quality Gateway' process used by requirements engineers to assess whether individual requirements identified for a system should be included in the final requirements specification (Robertson & Robertson, 2012)

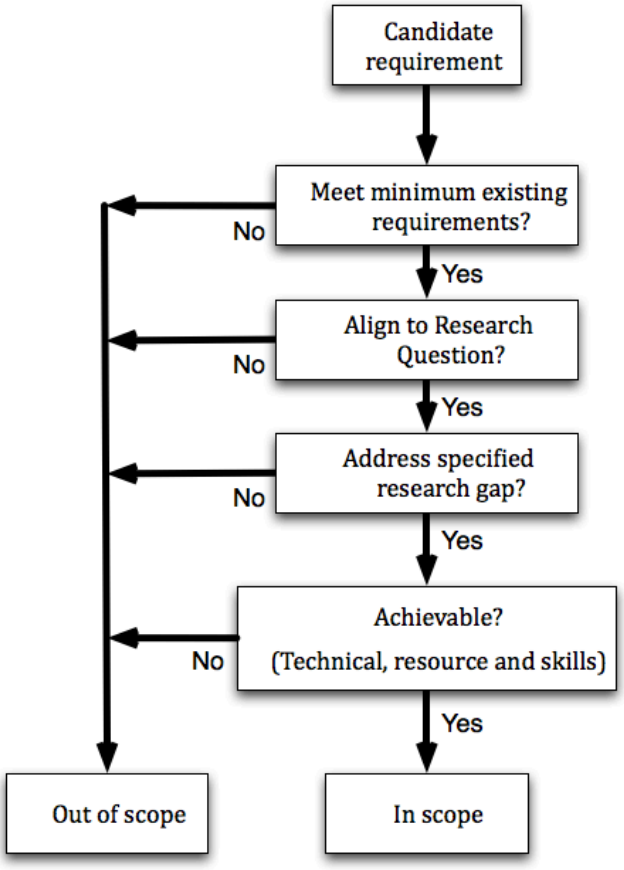

Figure 3-1 : Proposed requirements assessment methodology

By separating requirements as either external or internal, it facilitated the process of identifying those requirements that were more easily defined. The sections that follow will examine these external and internal requirements in more detail.

3.3.1 External requirements

The rationale applied to selecting what requirements to include started with the inclusion of what were deemed to be mandatory requirements, as indicated in section 3.2.2 (Legal) and 3.2.3 (Regulatory). Hence the requirements described in these sections were included in the proposed list.

Beyond these mandatory requirements, the technical recommendations outlined in section 3.2.1 were all included in the proposed list for a variety of reasons. The reasoning applied to each of these was as follows.

Use of Virtual Machines (VMs) were included as these were cited in Section 2.2.4 as having several benefits for malware analysis. In addition, by hosting these on a Linux-based host in an isolated network, the risk of malware escaping from the VM and migrating elsewhere is minimised (Pearce, Zeadally & Hunt, 2013). Furthermore, an implementation using VMs allows the testing environment to be scaled up to run multiple tests simultaneously. This means that larger quantities of data can be generated quickly.

Simulated network services were also included in the proposed requirements, as it is relatively easy to implement through open source software and provides the benefits outlined in section 3.2.1. Specifically, implementing simulated network services will provide an environment that maximises the observable activity of malware.

For similar reasons, the use of a vulnerable operating environment, as recommended in section 3.2.1, was included in the proposed list of requirements. As with the network service provision, this would provide a more fertile environment for malware to operate.

As indicated in Section 2.2, the research focuses on the evaluation of tools used to perform dynamic malware analysis. Of the two broad approaches to dynamic malware analysis (transition based and state based) identified in section 2.2.4, transition based logging was selected as it has the advantage of capturing more trace information, such as a file that is created and subsequently deleted between two machine states. Therefore, the tools that will be evaluated will be those that follow the Active Monitoring approach presented by Malin *et al.* (2008), see Figure 2-7.

Finally, the safe handling of malware recommendations cited in section 3.2.1 were included in the proposed requirements largely because these recommendations are aligned to the conditions of use for the VM environment available for this research.

The above external requirements are summarised in Table 3-2 below:

#	**Requirement**	**Rationale**
1	(Legal) Handling of malware and what it may access should be controlled.	See section 3.2.2 above
2	(Legal) Output of tested tool must be admissible.	See section 3.2.2 above
3	(Legal) Malware analysis tool output must be reliable	See section 3.2.2 above
4	(Regulatory) Novel methods must be validated	See section 20.1.5, Forensic Science Regulator (2016)
5	(Regulatory) The theory/technique should be peer reviewed or published	See section 20.16.1, Forensic Science Regulator (2016)
6	(Regulatory) Method should be a generally accepted	See section 25.2.3(e), Forensic Science Regulator (2016)
7	(Technical) Use a VM	See section 3.2.1 above
8	(Technical) Network service provision	See section 3.2.1 above
9	(Technical) Use vulnerable environment	See section 3.2.1 above

Table 3-2 : Proposed external requirements

Alongside the above externally set requirements, several internally set requirements were developed to facilitate the achievement of the externally set requirements.

3.3.2 Internal requirements

The internally set requirements were governed by the research gaps identified by the Research Question (see sections 1.2 and 3.1).

Pass/fail thresholds

Part of the evaluation of a software tool could be to assign a pass or fail threshold to a tool following a test, but this was rejected because it is not required by any external requirement. Furthermore, such a requirement is not part of the Research Question, see section 1.2. In addition, the general-purpose nature of the framework would be to apply different tools to the framework for testing, however each type of tool may have a different threshold level, making meaningful pass/fail comparisons difficult.

Also, given that there are no published pass/fail rates on any metric for any tool used for malware analysis that have been identified to date, deriving and justifying such a threshold would be difficult to defend and therefore a risk to the validity of the research. One manifestation of this could be that due to the lack of official guidance on the matter, the

acceptance threshold could vary between users. Another could be that if the level of the threshold were to change over time, the framework's relevance would quickly become dated.

Black box testing

Moving beyond the setting of thresholds to evaluate a software tool, the matter of how the tool is evaluated was considered to identify associated evaluation criteria. The use of black box testing (as discussed in section 2.3.1) is more viable than white box testing. This is due to the closed source nature of most of the software tools that are used by forensic practitioners and the time (and skills) that would be required to review source code.

Malware lab requirements

Having previously established that the research is to focus upon the more viable approach of evaluating tools used for dynamic malware analysis (see section 2.2.4), an initial requirement was to consider the construction of the lab used to perform the analysis. As discussed in section 3.2.1, there are no existing formal requirements for a technically valid malware analysis lab. Malin *et al.* (2008) offers some high level advice in terms of the environment itself, stating that a virtualised lab should be used (giving little consideration to alternatives). Elisan (2015) goes further and anecdotally suggests a malware lab used for dynamic analysis of malware should consider:

a) Analysis on both bare metal and virtual machines (VMs)
b) Observe how the malware behaves on different operating systems
c) Implement 'malware friendly' measures such as:
    i. Assigning administrator rights to the default user account
    ii. Disabling auto updates
    iii. Disabling User Access Control (UAC)
    iv. Setting the Internet browser to the minimum security level
    v. Install commonly exploited software
    vi. Creating honeypot files, eg: salaries.xls
d) Isolate the lab from the main network

The first of these (item [a]) was not fully adopted for this framework, as bare metal implementations are resource intensive and parallel processing is limited to the number of physical machines available (see section 2.2.4). Furthermore, the throughput of analysis is limited to the time it takes to restore the system to the pre-infection state. Instead a VM only approach is taken, as the resource for this is already in the research environment available. Furthermore, the ability to manage this remotely and in shorter timeframes renders a VM only

approach more practical for this research. The use of both platforms will be subject to further work (see section 7.4). The impact of this decision is that there is a risk to the validity of the results. This is, argues Martignoni *et al.* (2009), because when the malware is executed it may detect the "synthetic" environment. However, there is an increasing use of virtualised servers in modern I.T. environments and a study of 200,000 malware samples taken from 2012 to 2014, Wueest (2015) argued that only a small number of these cases detected it was running within a VM (see section 2.2.4).

In order to keep the scope of the research focused, the use of multiple operating systems (item [b]) was also not adopted. Although this is relatively easy to adopt, more recent operating systems implement tighter security controls that hinder the use of many of the security/malware analysis tools. This decision has little impact on the framework itself, as this is more of an implementation decision and is readily addressed by including additional VMs with disparate operating systems. As before, this is placed on the list of further work (see section 7.4).

The first four of Elisan's 'malware friendly' measures (item [c] i to iv inclusive) were all adopted into the framework as these are easy to implement and contribute to establishing a fertile environment for malware to activate.

The last two of Elisan's 'malware friendly' measures (item [c] v and vi) were not adopted into the framework, as the intention is to establish the minimum behaviour of any malware subjected to the framework. Furthermore, with regard to commonly exploited software (item [c] - v), not all users will have a given version of software installed or even at all. In addition, placing files with suggestive filenames, such as 'salaries.xls' (item [c] - vi) is reliant upon guessing what a malware binary is looking for on a host. The last of Elisan's recommendations (item [d]) has been adopted as it helps to address the legal requirements outlined in section 3.2.2.

Sourcing malware

To be as widely adaptable as possible, the framework was designed to accept malware from any source. Each source (where appropriate) can have an import module written to obtain local copies of malware binaries. For the purposes of this research a single source module linked to a feed provided by the website VirusTotal (2010) was used. This was done to simplify the import process and to provide a large number of malware binaries from the wild as quickly as possible.

Storing and handling malware

The storage of live malware for testing against software tools in the framework required a number of measures to be put in place. The university, for example, have set specific requirements to permit the storage of malware on their I.T. systems (see section 3.3.1). In addition, there was a need to minimise any cross-contamination between malware binaries; it is important that each malware binary cannot easily be accessed (or executed) by a user operating an implementation of the framework or by other malware binaries.

Conversely though, the ability to extract a malware binary, place it within the appropriate test area and execute it in an automated fashion was required for automated testing.

Finally, access to the library holding the malware binaries was restricted to a small number of users to minimise the risk of accidental or deliberate misuse.

Metrics

Section 2.3.2 identified several criteria that digital forensic practice is measured against. From these the Forensic Science Regulator's Codes of Practice and Conduct (2016) highlighted and discussed the following measures:

- Estimate uncertainty (Section 20.18 of the Codes)
- Traceability of reference data sets (Section 22 of the Codes)
- Validation of methods (Section 20.2 of the Codes)

An estimation of uncertainty is partly achievable in the form of a statistical confidence interval when comparing distributions of results from two tests conducted under different conditions. However, due to the complexity of calculating this measure, particularly when malware is involved, it was decided to not include this as a requirement within the scope of the PhD.

The traceability criterion is also difficult to address, as no existing standard and generally accepted malware corpora has been identified. The work of Garfinkel *et al*. (2009) has produced a corpus of realistic data, but this is not specifically tailored to housing malware for the purposes of testing malware analysis tools. However, a notable aspect of this research is that the framework was implemented and tested using a large population of real-world malware binaries (in excess of 350,000). This is relatively large when compared to other research groups who use fewer numbers of malware binaries and will be made available to others seeking to undertake research on the same dataset.

The last of these criteria (validation of methods) is partly viable within the timeframe for the research. Addressing the gap in the validation process identified in section 3.2.3, it is the validation of a software tool (and not the entire method surrounding its use) that is the focus of this research.

Validation is defined within the glossary of the Codes of Practice and Conduct (Forensic Science Regulator, 2016) as a means to demonstrate that a "method, process or device is fit for the specific purpose intended". Although not specifically mentioned, the meaning of 'device' could readily be applied to a software device or tool. However, it is not clear how such validation is performed or what measures should be used, e.g., accuracy, repeatability, etc.

One measure readily available is that of error, i.e.: the difference between the expected and observed values. Given the random nature of the data to be examined artefact values such as filenames are expected to vary much more than the quantity of artefacts produced each time a malware binary is executed. Hence, the framework should compare the quantity of expected and observed values, rather than the values themselves.

Validation of a tool measuring artefacts produced by malware is complicated by the fact that malware employs anti-forensic techniques to obfuscate the truth. Hence 'ground truth' is difficult to establish. The next best step is to compare what is reported by a tool against an independent and trusted source or 'oracle'. This will require the framework to (a) determine the expected value from an independent source and (b) be capable of retrieving the observed number of artefacts from a variety of tools applied to the framework for testing.

The internal requirements discussed above that have been included in the framework are summarised in Table 3-3. The internal requirements that were not included are summarised in Table 3-4.

#	**Requirement**	**Rationale**
1	Black box testing approach	Use of closed source software.
2	Malware lab requirements: • VM Only approach	Resource available, shorter test times, ease of automation and remote control.
3	Malware lab requirements: • Single operating system	Maximise results on single, older operating system.
4	Malware lab requirements: • Configure the OS to be malware friendly	Provide a fertile environment to provide 'best case' results for tools analysed.
5	Accept real-world malware from any source	Maximise the universality of the framework
6	Storing & handling malware: • Avoid cross contamination	Minimise risk to validity of results
7	Storing & handling malware: • Extract via automation	Facilitate automation of framework for tool testing
8	Storing & handling malware: • Restrict access to malware	Minimise accidental or deliberate misuse
9	Metrics: • Determine the expected quantity of artefacts from an independent source	In the absence of ground truth, provide an independent and authoritative measure to compare a tool against.
10	Metrics: • Read observed number of artefacts from a variety of tools under test	Focus upon quantities rather than values to counter anti-forensic approach of random values being used.
11	Metrics: • Validate tool by measuring difference of expected and observed numbers of artefacts	Provide measure of error. Provide an informed measure that addresses the confidence aspect of the Research Question

Table 3-3 : Proposed internal requirements

#	Excluded Requirement	Rationale for being out of scope
1	Pass/fail threshold	• Not part of the Research Question • Maximise general purpose aim of framework • No published thresholds to compare against • Could change over time, quickly dating the framework
2	White box testing	• Most commercial tools are closed source • Insufficient skills/time of practitioners to review code
3	Bare metal & VM environment	Regarding the bare metal side of this requirement: • Resource intensive (multiple physical machines) • Slow cycle time to reset machine between tests • Limited throughput in a given timeframe • No remote management capability
4	Multiple operating systems	• More recent operating systems have tighter security • Fewer malware analysis tools supported • Less fertile environment for malware to operate
5	Use commonly exploited software	• Anticipated that not all users will have a given version of exploitable software, reducing validity of tests
6	Use of honeypot filenames	• Assumes the malware is looking to harvest files • Requires guessing what the malware is looking for
7	Estimate of uncertainty	• Partially implemented in terms of statistical confidence • High number of variables, so too complex to calculate
8	Traceability	• No existing standard or malware dataset identified
9	Validation of method	• Partially implemented through validation of tool • Including validation of process/method requires skills in malware analysis techniques, which the researcher does not have

Table 3-4 : Excluded internal requirements

3.4 Analysis and design methodology

The Waterfall model (Royce, 1970) has been applied to the analysis and design of the solution to address the Research Question. The approach, argues Balaji and Murugaiyan (2012), works well where the requirements are clear beforehand. In the case of the MATEF, the requirements are reasonably fixed and clear (see section 3.3). Furthermore, the level of resources required to implement the model is minimal. This is particularly beneficial, as it was anticipated there would be little access to or response from the practitioner community on an on-going basis while the framework was under development. This would have been required if we used an alternative approach, such as the Agile development methodology (Collier, 2011).

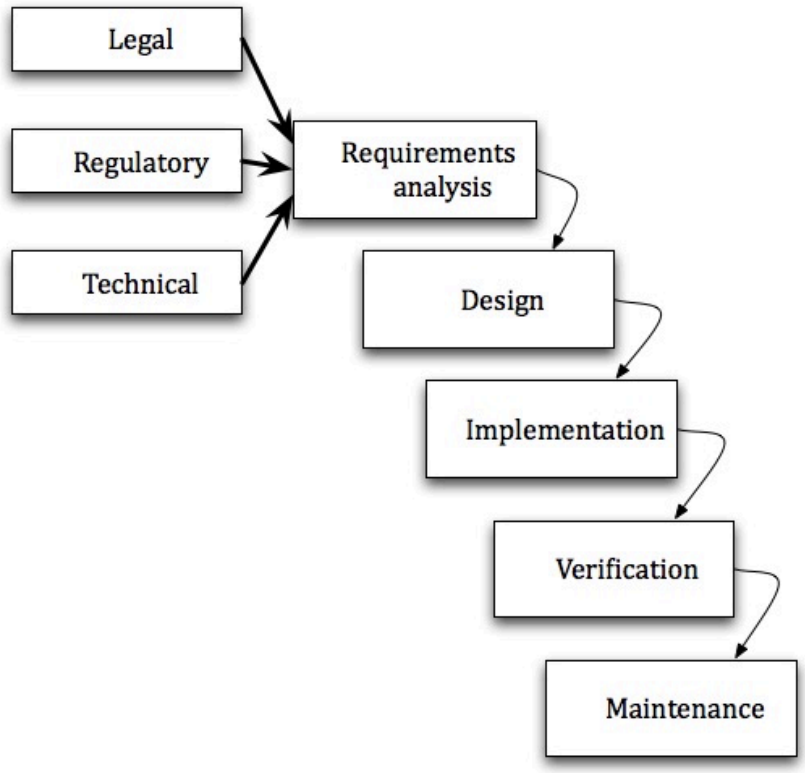

Figure 3-2 : Waterfall analysis and design model

3.5 Chapter summary

This chapter has identified two gaps in the field of malware forensics, namely: the lack of any definition of tool evaluation for malware analysis and the lack of any formal requirements for a technically valid malware analysis lab.

A review of existing technical, legal and regulatory requirements was explored and where feasible and relevant these have been adopted into the framework as a list of requirements set by third parties. We refer to these as external requirements.

A closer examination of the research question identified several requirements, which we refer to as internal requirements.

Collectively, both sets of requirements (external and internal) are chosen to both address the research question and set the scope of the PhD to keep it viable in the time and resources available. Hence, having identified these requirements, it is now possible to formulate a design and implementation for a Malware Analysis Tool Evaluation Framework, which is described in the next chapter.

Chapter 4 Designing and implementing a framework

In the previous chapter the requirements for a framework to test malware analysis tools were developed from the research question alongside a series of existing technical, legal and regulatory requirements. This chapter takes these requirements and translates them into a design for a framework named the Malware Analysis Tool Evaluation Framework (MATEF). Although requirements determine the constraints and minimum expectations of the framework, the aims of the framework define what it is to achieve. Hence, starting with the requirements identified in the previous chapter, section 4.1 takes these requirements and identifies a number of aims for the framework. The major components to achieve these aims are identified in section 4.2 before a discussion of their implementation is given in section 4.3. To evaluate how well the framework operates, the hypotheses are reviewed to assist in developing a testing strategy in section 4.4. This strategy is then used to inform the experiment design in section 4.5. Section 4.6 concludes the chapter with a discussion of the analysis strategy adopted.

4.1 Aims of the framework

Many tools can be used during the course of a malware investigation. Some of these tools make claims to be suited for malware analysis, while others do not. The MATEF aims to provide a mechanism to evaluate these tools by quantifying their ability to detect artefacts produced by real-world malware samples (see Aim 1, Table 4-1).

How such tools are employed for malware analysis is, according to Malin *et al*. (2008), subject to three broad analysis techniques: *temporal*, *relational* and *functional* analysis. Temporal analysis concerns the timeline of events surrounding reported activity, while relational analysis refers to the interaction between components of the malware and its environment. Finally, functional analysis relates to the actions the malware is reported to have performed.

Much of the temporal and relational analysis required with malware investigations can be achieved using conventional forensic analysis tools. It is the functional analysis that the MATEF sought to underpin by evaluating the ability of the tools used to detect the artefacts produced by the *behaviour* of malware (see Aim 2, Table 4-1). This behaviour typically manifests itself in the form of file, registry, process and network based artefacts.

Unlike regular software that is largely predictable, malware is typically unpredictable in nature and routinely implements anti-analysis methods. These methods include obfuscation

techniques designed to give misleading results under analysis. Hence mitigation against such risks should be considered when drawing conclusions obtained from tool testing using malware, see Aim 3, Table 4-1.

For the final research goal (see Goal 3, Table 1-2), the research needed to yield a software product that allowed a user to supply a candidate tool for malware analysis. The software product would then assess the candidate tool against pre-defined criteria; see Aim 4, Table 4-1. The results of this assessment aim to inform the practitioner's decision in the choice of tool used to perform malware analysis during a forensic investigation and provide quantifiable confidence in the reliability of the findings presented to a court of law.

Having identified the aims of the framework, consideration was then given as to how to achieve these aims. Hence, the following section seeks to identify the main components of the framework.

#	**Aim**
1	Use real-world malware
2	Evaluate a tool's ability to detect malware artefacts
3	Mitigate against anti-forensic techniques
4	Produce software product to test tools

Table 4-1 : Aims of the framework

4.2 Identifying & selecting the main components of the framework

One of the aims discussed in section 4.1 was for the framework to evaluate a software tool's ability to detect artefacts, and thus monitor malware behaviour (see Aim 2, Table 4-1). Hence the MATEF needed access to malware, the software tools to monitor the behaviour of such malware, a test environment suitable for executing the tools and malware and a management back-end to automate the whole process and record and analyse the results. Furthermore, in order to evaluate a given tool, a means of determining the expected number of artefacts for a given malware binary needed to be known (see Requirement 9 Table 3-3) and easily retrievable, ideally from a database.

Each of these elements is explored in the following sections, starting with the malware binaries themselves.

4.2.1 Malware sample source

In order to provide realistic results, the malware used to evaluate a given software tool needed to be real-world malware (a.k.a. malware 'in the wild'), as opposed to fabricated malware

(see Aim 1, Table 4-1). The stored malware employs password protected zip files to minimise contamination risk during handling (see Requirements 6 & 8 Table 3-3).

To satisfy the requirement to work offline (see Requirement 1, Table 3-2), malware obtained from any source needed to be imported locally to and stored in a malware library.

4.2.2 Malware library

In addition to satisfying the need to work offline (thereby providing readily available copies of the malware) and to simplify automation, each malware binary was to be accessible through a consistent file naming convention (see Requirement 7 Table 3-3). Also, in line with these requirements, access to this library was restricted to authorised users of the framework only.

In addition to the malware binary file, information on its expected behaviour also needed to be stored locally as well (satisfying requirement 9 from Table 3-3). To be made readily available, this information was stored in a malware database.

4.2.3 Malware database

The malware database needed to contain details of each malware binary held in the malware library (see section 4.2.2). As a minimum, the details stored included the hash value of the binary and the number of artefacts created as a result of creating, modifying or deleting files or registry keys. In addition, the number of ports opened and processes spawned as a result of executing the malware were also stored.

To facilitate stratification of the data, the database stored Boolean properties of each malware binary (where available), such as whether the malware configures itself to start automatically upon boot or if it disables anti-virus software.

The database itself was to be open source to ensure it is readily deployed with the framework and can be built and managed using automated scripts. The management of the database, including the importing of malware and the testing of software tools was to be controlled by management scripts to facilitate automated testing across many malware binaries.

4.2.4 Manager scripts

The manager scripts were to perform two fundamental roles, namely overseeing the testing of software tools and the interaction with the malware database itself.

The first of these required a script to initially construct the database tables and perform basic database management operations. This included the capability to import details of the

artefacts produced when executing malware and to retrieve them during subsequent analysis on provision of an identifier, such as a hash value.

The second fundamental role was to enable a user to initiate a test that operates and manages a bank of virtual machines (see Requirement 7, Table 3-2). The script was to also automatically execute the tool under test and then the malware for a specified length of time before resetting the virtual machine (see Requirements 2 & 5, Table 3-3). Following a given test the management script was also to extract a given tool's log file and write it to a specified location in a consistent and standard format (to facilitate subsequent analysis), regardless of the original log file format.

These two roles are dependent on two additional components, namely an independent source of malware behaviour data (referred to as the 'Oracle') and an environment within which to test the software tools. The former is discussed in the next section (4.2.5), whilst the latter is addressed in section 4.2.6.

4.2.5 The Oracle

Due to the lack of any theoretical or easily determined 'ground truth', the MATEF needed to determine the expected quantity of artefacts from an independent source (see Requirement 9, Table 3-3). The random nature of the data (malware) is such that the reported expected value is little more than an approximation of the 'ground truth'. This source, referred to as the 'Oracle' could be conceivably be any one of a number of online environments, such as Anubis (2010), F-Secure (2011) and ThreatExpert (2011) (see Table 4-5 for a more comprehensive list). The ability to determine the number of expected artefacts for a given malware binary when it is executed was the main requirement; see Requirement 9 in Table 3-3.

An important point to make here is that MATEF's purpose was to evaluate analysis tools and not to submit new or 'zero-day' malware to any of these sandboxes. Malin *et al.* (2008) point out that files submitted to such systems may be automatically shared with other vendors and third parties. The impact of this is two-fold: First, an investigator may be submitting a malware sample that is targeted to the victim. The impact of this is that hard-coded details such as usernames, passwords, or internal IP addresses may be inadvertently distributed. Secondly, the attacker who planted the malware will likely be alerted to the discovery and change their tactics. Hence the use of such sandboxes for live investigations may not be deemed an acceptable risk. Another significant problem with calling upon third-party sandboxes to identify malware behaviour is the lack of control the investigator has over the conditions under which the malware is executed.

In light of the above, the MATEF provided two key benefits for the forensic investigator: the ability to make an informed decision on which tool to use to perform offline malware analysis and the ability to customise the test environment to evaluate how the tool performs under different conditions, e.g.: operating systems and execution time.

With the source of Oracle information in place providing details on the expected behaviour of a malware binary, it was down to the tool under test to establish what behaviour can be observed executing the binary. To do this a safe and controlled environment needed to be provided in order to operate the tool and the malware.

4.2.6 Test environment

The test environment of the MATEF is one that will need to be managed via an automated script and have sufficient capacity to enable multiple tests to be run in parallel. In this way the data collection capacity of the MATEF will increase, helping to reduce the time required for large scale tool testing.

The anticipated variability of the malware under analysis may impact on the statistical power of the results (Smith, 2012). Hence, by increasing the number of malware binaries analysed from the library the statistical power (and hence the statistical significance) of the results should increase.

Closely linked to the test environment is the Internet simulation component, providing a networked environment containing common network services.

4.2.7 Internet simulation

The provision of network services (see Requirement 8, Table 3-2) provides the MATEF with an added level of realism to malware running within the Test Environment. Bayer *et al*. (2009) report that over 45% of malware they examined engaged in TCP traffic, which is not possible without an endpoint to initiate a connection to.

It is important this network provision is simulated to minimise any risk of the malware stealing any data or committing any unauthorised access to other networks (see Requirement 1, Table 3-2). Requests and responses are passed to and from common network services that are exposed to the test environment through the component.

A significant product of the test environment (assisted with the Internet simulation) is the log file from a given tool under test. To form any conclusions on a given tool, the log file it produces must be analysed.

4.2.8 Analysis component

In order to undertake analysis of a software tool, the analysis component needed to establish four things. The first of these was to establish what the tool is to be compared against. Previously, it was argued this should be the expected quantity of a given artefact, as opposed its value (see section 3.3.2). In this research, this is referred to as the *Expected value*. This value needed to be determined by an independent source (see Requirement 9, Table 3-3).

Secondly, the analysis component needed the capability to extract the number of artefacts observed (referred to as the *Observed value*) by the tool under test from a log file bearing a filename that can be determined programmatically, thus allowing multiple log files from different VMs and tests to coexist (see Requirement 10, Table 3-3).

A third analysis requirement was that the analysis components must establish an assessment of the difference between the Expected and Observed values (see Requirement 11, Table 3-3). This is a critical value and subject to the aims of a given test, forms the basis of the comparison between tools or multiple executions of the same tool to evaluate repeatability.

The final analysis requirement was a structured test design that was informed by one or more hypotheses that determined the aim of the analysis.

Test Design

As discussed previously (section 4.2.6) the malware to be studied using a given tool was anticipated to be highly variable. In order to isolate any observed changes as a result of a test control measure over variability of the malware itself, the analysis process needed to separate the malware selected for test runs into two groups. The first of these groups contained a list of the malware that exhibit variability in the numbers of artefacts observed when run under the same conditions. The second group would comprise a list of the malware that produced the same number of artefacts when run under the same conditions, and is hence repeatable. It was anticipated that any subsequent analysis would then focus upon the latter group to effectively filter out false positives in the data.

From a legal perspective, the overall aim of the analysis was to identify software tools that were ‘reliable’ (see Requirement 3, Table 3-2). Although open to interpretation, this can be pinned down a little more if the regulatory requirement for validation is also considered (see Requirement 4, Table 3-2). The focus of this research was to validate a software tool as part of the process of validating a method. The difference between the Expected and Observed values was selected as the metric for this analysis, as discussed in section 3.3.2.

Online sandboxes for malware analysis typically offer no control over the parameters of the test, such as the length of time that the malware is executed. This parameter is one that was easily adopted as a control measure for the test process to determine if different execution times produce more or fewer artefacts. If a practitioner can determine an execution time beyond which there is little added benefit to their findings, this would save valuable analysis time. Furthermore, the ability to quantify the impact of different execution times on the findings provides information to the practitioner where previously there was none. Hence, the hypothesis to determine the impact of the execution time is presented in Table 4-2:

$H_{1.0}$	Changing the execution time of malware has no significant impact on the number of malware artefacts observed by a given tool.
$H_{1.1}$	Changing the execution time of malware has a significant impact on the number of malware artefacts observed by a given tool.

Table 4-2 : Hypotheses 1 – Does changing the execution time affect how many artefacts are observed?

As well as the question of how long to run a tool for before concluding no further artefacts will be observed, the practitioner will seek to justify their choice of tool to the court. Hence, the hypothesis to determine which (if any) of two tools is able to detect a greater number of artefacts under the same operating conditions is presented in Table 4-3:

$H_{2.0}$	There is no significant difference on the number of malware artefacts observed by Tool A when compared to Tool B, under the same conditions.
$H_{2.1}$	Tool A is able to detect a significantly greater number of artefacts when compared to Tool B, under the same conditions.
$H_{2.2}$	Tool B is able to detect a significantly greater number of artefacts when compared to Tool A, under the same conditions.

Table 4-3 : Hypotheses 2 - Which tool observes more artefacts?

Figure 4-1 shows how the components described above are configured into the MATEF, together with the information flows between components. Note boxes in grey are external components that sit outside the MATEF. At present the statistical analysis component is performed using an independent statistical analysis tool. It is envisaged that future development of the MATEF will include a statistical component within the MATEF.

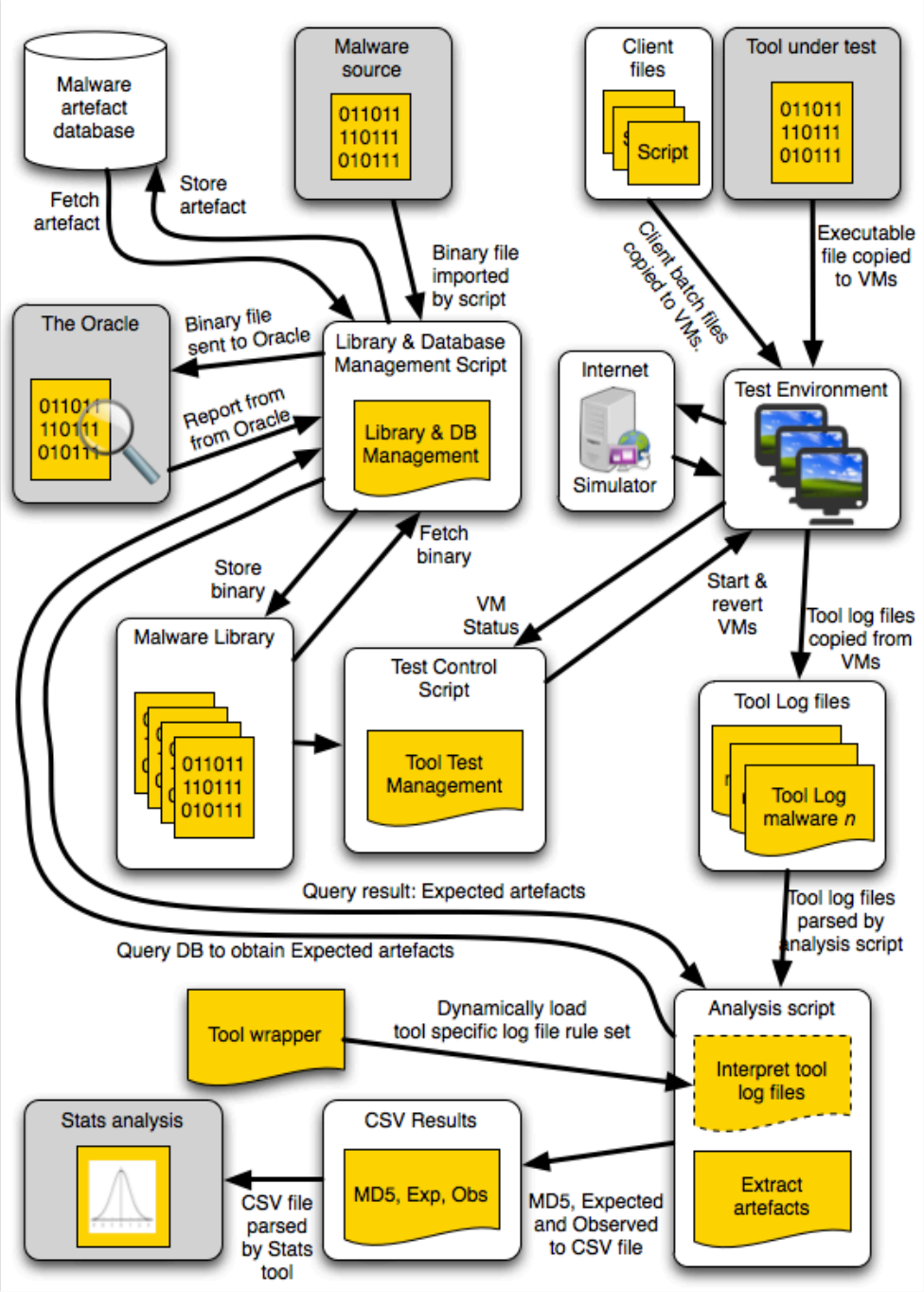

Figure 4-1 : MATEF components

Given that statistical analysis is currently performed outside of the MATEF, the analysis component produces an output that can be analysed statistically by third-party software.

In order to test and hence evaluate the MATEF design an implementation was undertaken, as discussed in the next section.

4.3 Implementing the MATEF framework

4.3.1 Malware sample source

Referring to requirement 7 in Table 3-3, the MATEF should be capable of accepting malware from a variety of sources. Elisan (2015) identifies several sources from which malware may be freely obtained. These were considered as a source for the MATEF, but were rejected on the basis that the numbers of binaries available from these sources are relatively small and not easily extracted in large numbers. Enthusiasts typically run such sources on a voluntary basis, resulting in sporadic support.

Other sources of malware include Honeypots (Gashi, Sobesto, Stankovic & Cukier, 2013), but this approach was again discounted on the grounds that it takes time to build a large collection of samples. Furthermore, a solution for the MATEF is sought that minimises the effort on the part of data collection. The requirement to build and commission a honeypot to initially gather malware binaries may discourage others from adopting the MATEF.

A more viable approach was offered through contact with security research organisations, such as VirusTotal (2010). VirusTotal provide a mechanism to feed malware submitted to their scanning platform through a specified email address. Each email contains a single malware file attachment encrypted in a password protected zip file bearing a filename matching the file's MD5 hash value. The use of VirusTotal as a source conveniently satisfies requirements 6, 7 and 8 from Table 3-3. Malware delivered via email attachments to the MATEF in this way is then extracted and stored in the Malware Library.

4.3.2 Malware library

The simplest approach to storing the malware binaries was to store them in a folder structure on disk with access permissions set to limit access to the files by unauthorised personnel. Using hash values as filenames to identify the malware, each file could also be encrypted with a password to both limit access and minimise accidental or deliberate cross-contamination. Malware located in the library was periodically sent to the Oracle (see section 4.3.5) where the results returned were then stored in the Malware database.

4.3.3 Malware database

Implemented in SQLite (https://sqlite.org/), the Malware database is comprised of the tables represented in Figure 4-2 below.

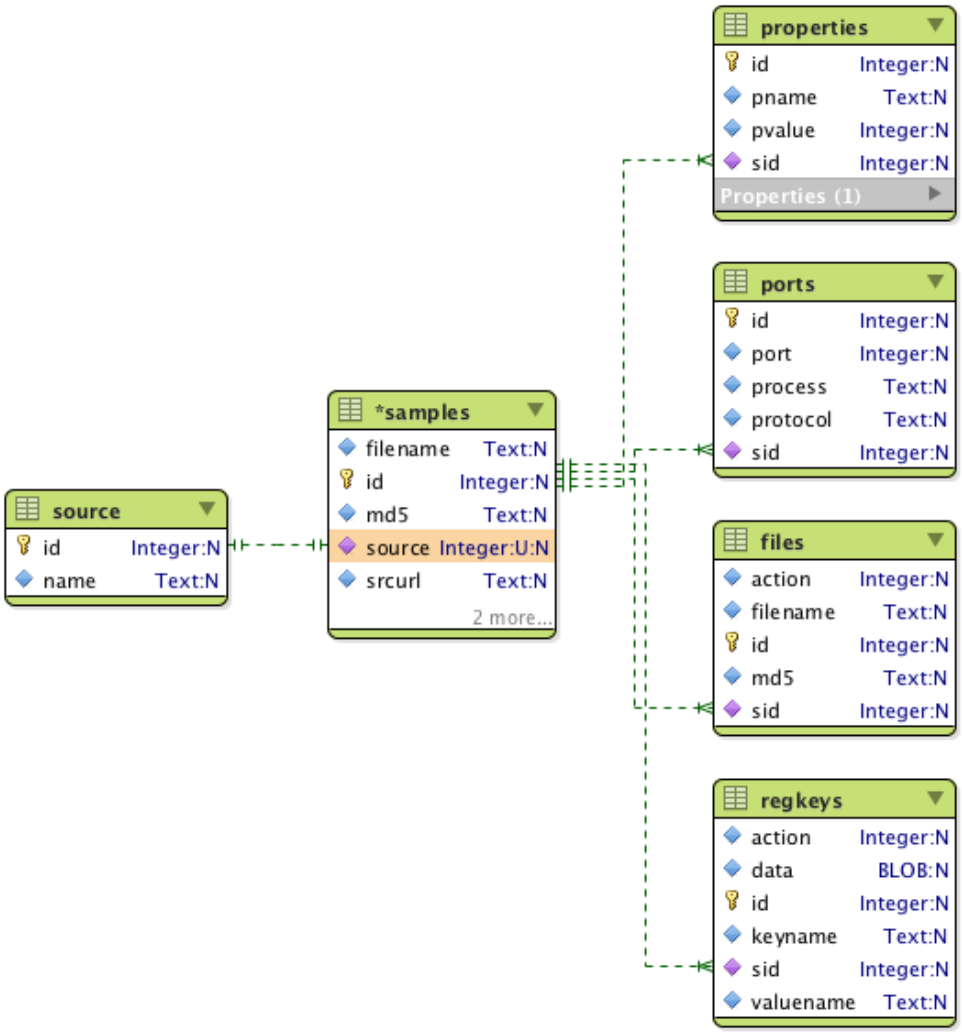

Figure 4-2 : MATEF Malware Database

The database schema follows a relational database approach to model the artefacts associated with malware. The rationale for organising the tables into ports, files and registry keys is that the framework's aim is to evaluate tools used to detect the artefacts that typically manifest in the form of file, registry, process and network based artefacts (see section 4.1). By organising the artefacts into these groups (tables) it simplified the process of selecting appropriate malware binaries to a given tool that may, for example, only be designed to detect port activity. Each of the fields is summarised in Table 4-4.

Table	Description of table	Field	Description of field
source	Lookup table of Oracle sources	id	ID of malware source
		name	Name of malware source
samples	List of all malware kept within MATEF	id	ID of malware binary
		filename	Full path to encrypted (ZIP) binary file in library
		md5	Hash value of unencrypted binary file
		source	ID of Oracle source
		srcurl	URL to Oracle report for binary
properties	List of properties associated with each binary, eg: ability to start on Windows boot, disable anti-virus, etc	id	ID of property
		pname	Property name
		pvalue	Property value
		sid	Foreign key: ID of malware binary (samples.id)
ports	List of all ports opened by malware	id	ID of port entry
		port	Port number
		process	Name of process linked to port
		protocol	Protocol used
		sid	Foreign key: ID of malware binary (samples.id)
files	List of all files linked to the malware	id	ID of file entry
		action	Action perfromed, eg: open, create
		filename	Name of file
		md5	Hash value of file
		sid	Foreign key: ID of malware binary (samples.id)
regkeys	List of all registry entries linked to the malware	id	ID of Registry key entry
		action	Action perfromed, eg: open, create
		data	Data for newly created value (if any)
		keyname	Registry key name
		valuename	Newly created value under key name (if any)
		sid	Foreign key: ID of malware binary (samples.id)

Table 4-4 : MATEF database field list

The one-to-many relationship between the *samples* table, the *properties* table and each of the artefact tables (*ports*, *files* and *regkeys*) meant that when a new malware binary was added to the database multiple properties and artefacts could be recorded against a given binary. The design is extensible in that if a new artefact group were to be added, eg: remote IP addresses that are contacted, then a new table with the associated fields may be added to accommodate this new group.

As previously mentioned in section 4.2.3, the management of the database was controlled by management scripts to import new samples and to facilitate automated testing across many malware binaries.

4.3.4 Manager scripts

The manager scripts were implemented in Python as there is an extensive library of existing code available to build upon, such as provided by Ligh *et al.* (2010). The three main scripts are as follows:

dbmgr.py — The database manager script, builds on code developed by Ligh *et al.* (2010). The script manages the import of new malware binaries, the storage and retrieval of malware aretfacts and files.

pms.py — The Program Manager Script (PMS) divides the test between multiple VMs, each independently managed by a management script (mgr.py, see below). On start up, PMS determines if previously used hashes have been requested (for repeatability testing) or if new, randomly selected ones belonging to an artefact type group are to be used, see Pseudocode 1.

mgr.py — Multiple instances of this script (see Pseudocode 2) are generated by the PMS (see above) to oversee the operation of a single VM. Each of these instances parses a hash list file associated with the VM, identifying a hash on each pass. Artefacts associated with this hash are identified from the malware database and stored in files for later analysis. The script then copies batch files to a network share visible to the VM, which is then booted. The batch files control the operation and timing of the tool under test. On completion the tool's log file is copied to network share before the VM is reverted in readiness for the next test.

```
============================================================
PMS.PY
Arguments: ArtefactTypes, HashListFolder, NumBins

============================================================

vmFirst    = 1   // Number of first VM to use
vmLast     = 60 // Number of last VM to use

// Get list of hashes for test
// If HashListFolder is null, pick new hashes
// Otherwise, uses hash files from supplied folder
IF HashListFolder == ""
      // Randomly select NumBins hashes, based on Artefact
      HashList = GetSamples(ArtefactTypes, NumBins)

      // Create separate hash list files, one for each VM
      CreateHashListFiles(HashList,vmFirst,vmLast)
ENDIF


// Divide the testing between multiple VMs
FOR vmNum = vmFirst to vmLast

      // Filename of hash list file
      hlFile = HashListFolder + "hashList-vm" + vmNum

      RUNSCRIPT 'mgr.py'
        WITH vmNum, hlFile, ArtefactTypes

      // Stagger the VM startups to minimise load
      Sleep 10 seconds
ENDFOR


//Wait for all VMs to finish running
VMRunning = Number of VMs running
WHILE VMRunning > 0
      VMRunning = Number of VMs running
ENDWHILE

// Preserve results
COPY ToolLog files from TestFolder to DataFolder
COPY List of hashes used for test to DataFolder
```

Pseudocode 1 : Program Manager Script (PMS.PY)

```
============================================================
MGR.PY
Arguments: vmNum, hlFile, ArtefactTypes
============================================================
// Set path to a network share specific to the vmNum
BaseFolder  = A network share location
ShareFolder = BaseFolder + "/" + vmNum
Count        = 0
FOR each hash in hlFile
      // Store artefacts of used hashes for later analysis
      IF ArtefactTypes includes REGISTRY artefacts
            regArtefactList = Registry artefacts for hash
            Filename = vmNum+Count+"regArtefacts"
            WriteToFile(Filename, regArtefactList)
      ENDIF
      IF ArtefactTypes includes PORT artefacts
            portArtefactList = Port artefacts for hash
            Filename = vmNum+Count+"portArtefacts"
            WriteToFile(Filename, portArtefactList)
      ENDIF
      IF ArtefactTypes includes FILE artefacts
            fileArtefactList = File artefacts for hash
            Filename = vmNum+Count+"fileArtefacts"
            WriteToFile(Filename, fileArtefactList)
      ENDIF

      Path = GetSamplePath(hash) // Get library path to MW

      Copy Client batch files from Library to ShareFolder
      Extract malware binary from Path to ShareFolder

      Start VM number vmNum
      // Wait for VM to complete booting or timeout
      // Returns 0 if complete or -1 if timed out
      Result = CALL WaitVmStart()
      IF Result == 0
            CALL WaitVMStop()     // Wait for shutdown
      Revert the VM
      Rename Tool Log file to include the VM & Test number
      Copy ToolLog file from VM share to TestFolder
      Clear files from VM share
      Increment Count
ENDFOR
```

Pseudocode 2 : Manager Script (MGR.PY)

4.3.5 The Oracle

Two of the proposed requirements for the MATEF that inform the choice of Oracle were the need to automate the process (see Requirement 7, Table 3-3) and the ability to determine expected quantity of artefacts (see Requirement 9, Table 3-3). Thus, the former required the Oracle to accept automated submissions of malware to its system as well as providing an interface to programmatically collect the results. Furthermore, the report produced by the Oracle needed to be of a form to facilitate parsing by a script so the findings could be imported into the malware database. The latter requirement meant that the Oracle should report an indication on what it observed, thereby enabling the quantity of observed artefacts to be calculated.

Further to the above requirements of the MATEF, the design of the database (section 4.3.3) meant that the Oracle report should also include artefacts identified as either file, registry, network or process related artefacts (see Figure 4-2). In addition, the design of the database suggested that the actions performed on these artefacts should also be reported.

In selecting a sandbox source for use as the Oracle in this research implementation of the MATEF, the sandboxes listed in Table 4-5 were considered.

The ThreatExpert (2011) sandbox was initially considered as an Oracle source for the MATEF, but repeated reliability issues at the time were considered a threat to the progress of the research. Like ThreatExpert, the Anubis (2010) sandbox also provided a readily available and convenient interface to upload multiple malware samples via a script. The ability to add a new module to interface to a given sandbox demonstrates the flexibility of the MATEF to use more than one source for the Oracle (see the database table named *source* in Table 4-4). This is important as online sandboxes can be transitory in nature and therefore not guaranteed to be available in the future.

Name	Description
Anubis (2010)	Online malware analysis system. Enables individual executable files to be uploaded via a web form or bulk quantities via FTP. Results are sent to a designated email address.
Comodo Valkyrie (Comodo Group, n.d.)	Provides 'File Verdict Service' employing different methods to analyse a given file.
F-Secure (2011)	Online malware analysis system. Enables executable files to be uploaded via web form or uploaded via FTP. Files sent via FTP must also have an associated text file uploaded via the web form, limiting the ability to automate.
Joebox (Joe Security, 2017)	Commercial online malware analysis system with a free license option. Uses filtered Internet access to malware under analysis. Free account license does not allow different environments (eg: operating systems) to be specified and is limited to 10 submissions per day.
Malwr (Malwr, 2016)	Online system that uses the offline Cuckoo (2016) sandbox environment. No option to configure test environment. No facility to automate submissions available.
Payload Security (Payload Security, n.d.)	Online malware analysis system that can be purchased for offline analysis of large quantities of data. No bulk analysis is available for online platform. Environment and even user actions can be scripted for analysis.
ThreatAnalyzer (ThreatTrack Security, 2016)	Online malware analysis system, formally run operated as the academic programme known as 'CWSandbox'. Now a commercial, subscription service.
ThreatExpert (2011)	Online malware analysis system. Enables individual executable files to be uploaded via a web form. Early trials with this as an Oracle source found it to be unreliable at delivering reports.

Table 4-5 : Online malware analysis sandboxes

4.3.6 Test environment

The research environment provided a *VMWare* (VMWare, 2016) resource, available to enable multiple virtual machines (VMs) to be operated remotely and via a scriptable interface. This allowed for the deployment of multiple VMs, each meeting the need for the use of a VM in the MATEF (see Requirement 7 in Table 3-2 and Requirement 2 in Table 3-3).

To provide as fertile an environment for the malware as possible (to satisfy Requirement 9, Table 3-2 and Requirement 4, Table 3-3), the operating system needed to be subject to a number of vulnerabilities. Sikorski and Honig (2012) recommend Windows XP for this reason. Thus, this was selected as the operating system for the MATEF implementation.

Each VM needed to be configured to operate on a closed network with no Internet access (see Requirement 1, Table 3-2). However, to meet Requirement 8 of Table 3-2 (network service provision) simulated Internet services needed to be available as well.

4.3.7 Internet simulation

Malware typically exploits one or more Internet based protocols such as *Hypertext transfer protocol* (HTTP) (Traore, Awad & Woungang, 2017, p. 2), *Domain name system* (DNS) (Wang, Lin, Cheng & Chen, 2017) and *Internet relay chat* (IRC) (Angrishi, 2017) to exhibit more behaviour post-infection of the host computer. Tools such as *ApateDNS* (FireEye, 2017) used for responding to DNS queries and *MockServer* (Bloom, 2017) used for responding to HTTP requests can be used to provide a simulated Internet environment. However, the disparity of these tools makes it harder to manage them collectively as a single entity in a test environment.

A more integrated solution known as *iNetSim* (Hungenberg & Eckert, 2016) has been selected for inclusion in the MATEF as the Internet simulator. Hungenberg and Eckert describe the tool as "a software suite for simulating common internet services in a lab environment, e.g. for analysing the network behaviour of unknown malware samples". The tool provides simulated services for several services including HTTP, Simple Mail Transfer Protocol (SMTP) and File Transfer Protocol (FTP).

4.3.8 Statistical analysis

Previously, it was identified that, for a given malware binary, the Analysis component (see section 4.2.8) would need to establish (a) the expected quantity of artefacts; (b) the observed number of artefacts and (c) the difference between the expected and observed values.

These requirements were most readily made available via a script that meets the requirements of (a) and is able to interrogate the malware database (via the management script). To address (b), the script examines the raw log files produced by a given tool under test. Given the disparity in the formats of log files from different tools, functionality such as identifying and counting the number of network ports opened by a tool has been abstracted in the script. The specifics of interpreting a given log file format were implemented in a separate script (termed a *wrapper*) that forms a plug-in for each tool under analysis. Thus, to facilitate

extensibility, the introduction of a new tool requires only that the smaller wrapper script is produced for that tool whilst the functionality and logic of the process as a whole is held in a single master script (*analyseMATEF.py*), see Pseudocode3 below

```
=========================================================
analyseMATEF.PY
Arguments: logFileFolder    // Folder containing log files
           analysisType     // Type of analysis to perform
=========================================================

BASEDATA   = A network share location for all results
CSVFNAME   = "analysisCSV.csv"
LOGPATH    = BASEDATA + logFileFolder
CSVPATH    = BASEDATA + logFileFolder + CSVFNAME

vmFirst    = 1   // Number of first VM to use
vmLast     = 60 // Number of last VM to use

csvFile = createCSVFile(CSVPATH)

// Get the Log file from each VM used
FOR vmNum = vmFirst to vmLast

      // Call readTooLogFile, returning 3 values:
      //    Res.hash         MD5 hash of binary file
      //    Res.Expeceted    Expected number of artefacts
      //    Res.Observed     Observed number of artefacts
      Res = readTooLogFile(logFileFolder, vmNum, analysisType)

      WriteToCSV(csvFile, Res.hash, Res.Expected, Res.Observed)
ENDFOR


-----------------------------------------------------------
FUNCTION readToolLogFile(logFileFolder , vmNum, analysisType)

SWITCH analysisType:
      CASE 1:
            // Examine each log file for a specified VM
            // Return the MD5 hash, expected and observed
            // number of files created by the malware
            Result = analyseFiles(vm, logFileFolder)
      CASE 2:
            // Examine each log file for a specified VM
            // Return the MD5 hash, expected and observed
            // number of ports opened by the malware
            Result = analysePorts(vm, logFileFolder)
      // etc.
ENDSWITCH

RETURN Result
```

Pseudocode 3 : Analyse MATEF Script (anayseMATEF.PY)

Regarding the difference between the expected and observed values (item (c) above), it is envisaged that at a later date this script will also undertake statistical analysis on this data and report its findings from simply running the script. However, given the constraints of the research, this analysis is currently undertaken using external statistical tools, such as SPSS (http://www.ibm.com/spss).

4.4 Testing strategy

The discussion on test design (section 4.2.8) identified two hypotheses (see Table 4-2 and Table 4-3) that address the aims of this research. Taking each of these as a control measure, the following statistically independent variables (IV) were identified:

IV_1	Execution time
IV_2	Analysis tool

Table 4-6 : Independent variables

IV_1 Execution time

One item of metadata provided by the Oracle (see section 4.2.5) is the execution time for which each instance of malware is executed. As discussed in section 4.2.8, variation of the execution time may result in a variation of the number of artefacts produced by the malware. Depending on the design and aims of the malware, different artefacts can be produced at different times. It is impractical to expect to observe every single artefact that might be produced by the malware, thus the focus here is to determine the number of initial artefacts created by the malware.

The Oracle reports that the execution time varies between each malware binary, but has an average value of around 5 minutes. The strategy therefore, was to deviate from this execution time by both increasing and decreasing the length of time each malware binary is run to address Hypothesis 1 (see Table 4-2).

IV_2 Analysis tool

Two tools commonly used to study port activity on a computer under investigation are Process Monitor (Russinovich, 2016) and TCPVCon (Russinovich, 2011), the command-line version of the TCPView tool.

These two tools were selected to address Hypothesis 2 (see Table 4-3), as they can both be controlled from the command-line. The framework is designed to work with analysis tools that operate on a Windows operating system and are capable of being managed via a command-line interface. Hence the ability to start and terminate the tool via the command-line as well as the ability to programmatically export a log file of the observed artefacts was required. Therefore, the testing strategy chosen is to compare the number of artefacts detected by two software tools that meet the above requirements.

With the testing strategy in place, the design of the experiments could then be formulated.

4.5 Experiment design

The MATEF implementation used for this research has 60 identical VMs available that can operate in parallel. A total of eighty (80) malware binaries were applied to each VM, giving a total test space of 4,800 (60 x 80) malware binaries.

These malware binaries were initially selected at random from a subset of malware binaries in the Malware Library that exhibit some form of network activity. This initial random dataset was then applied to each of two tools for 1 minute, 5 minutes and 10 minutes and 10 seconds to explore the impact of this variation around the reported average execution time of the Oracle (see discussion on Execution time in the previous section).

Early testing of the data collection process identified that a number of malware binaries produced highly variable numbers of artefacts on each execution. To minimise the impact this 'random noise' had on the objective to measure a given tool's ability to detect malware artefacts, each execution time was repeated twice. This lead to the creation of three datasets of observations for a given tool; each for the same length of execution time. Any malware binary that did not produce the same observed value in all three datasets was then filtered out, leaving a dataset of observations of malware behaviour that is considered repeatable. This decision was made to minimise any error resulting from executing the malware and to improve the repeatability of the process, thereby aligning it more to a scientific methodology (see Table 1-1). Whilst it is acknowledged that the impact of this decision is to evaluate a tool on only a subset of malware samples, it is argued that this does not impact on the validity of the framework, which remains unchanged. However, this issue is included in the discussion on the limitations of the research (see section 6.5).

As discussed earlier, Process Monitor and TCPVCon were applied to Hypothesis 1 (impact of execution time, see Table 4-2) and Hypothesis 2 (comparison of one tool with another, see

Table 4-3), as indicated in Table 4-7. Each Test was comprised of two data collection runs, each running for different lengths of time.

Experiment	Test	Description	Hypothesis
Experiment 1 Comparing Process Monitor at different execution times	1.1	Comparing 1 minute to 5 minute execution times	1
	1.2	Comparing 1 minute to 10 minute execution times	1
	1.3	Comparing 1 minute to 10 second execution times	1
Experiment 2 Comparing TCPVCon at different execution times	2.1	Comparing 1 minute to 5 minute execution times	1
	2.2	Comparing 1 minute to 10 minute execution times	1
	2.3	Comparing 1 minute to 10 second execution times	1
Experiment 3 Comparing two tools at the same execution time	3.1	Process Monitor vs TCPVCon, run for 10 seconds	2
	3.2	Process Monitor vs TCPVCon, run for 1 minute	2
	3.3	Process Monitor vs TCPVCon, run for 5 minutes	2
	3.4	Process Monitor vs TCPVCon, run for 10 minutes	2

Table 4-7 : List of Experiments

With the experimental design established, the analysis strategy was then considered.

4.6 Analysis strategy

The previous section established the design of the experiments to be run. Before considering how to analyse any results obtained through this method it was necessary to gain a clearer understanding of the nature of the data to be collected and analysed, as this informed the choice of applicable analysis methods available.

4.6.1 Describing the data to analyse

Stevens (1946) identified the relationship between what is being measured and the numerical values they represent. This has since developed into what is commonly known as the *Levels of Measurement*, see Figure 4-3.

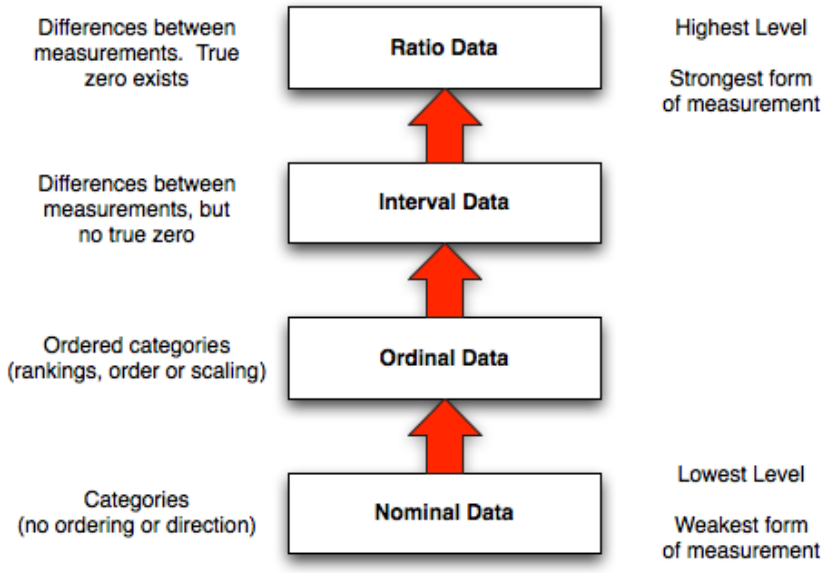

Figure 4-3 : Levels of measurement

Each level inherits the properties of the preceding, meaning it is able to accommodate the type of data of the levels below it. Furthermore, each level has associated with it a number of valid operations, see below.

Level	Examples	Operations
Nominal	Port number, Filename, Registry key name	=, <>
Ordinal	Threat level of malware (eg: Low, Medium, High)	=, <>, <, >
Interval	Number of ports opened by malware	=, <>, <, >, +, -
Ratio	File size of malware	=, <>, <, >, +, -, *, /

Table 4-8 : Measurement levels

Under this scheme, the port number of each opened port on a computer and the name of a file or registry key created are all examples of *nominal* data, based on the properties identified by Panik (2005). The names and the number (in the case of port numbers) are nothing more than labels that refer to the artefact it represents. There is, for example, no inherent difference between the network port 80 and port 443. Although the former is commonly used for unencrypted web browser traffic and the latter for encrypted traffic, there is little else that can be determined from comparing them. To state that one is greater than the other is or there is a 'difference' of 363 between them is meaningless, as the port numbers do not represent a quantity.

Both Hypothesis 1 (Table 4-2) and Hypothesis 2 (Table 4-3), examine 'the number of malware artefacts' to determine an outcome. Hence, although the individual malware artefacts are nominal in nature, a count of their numbers represents a quantity.

The quantities of artefacts produced by different malware binaries can be compared not just in an ordinal fashion (eg: one binary produces more artefacts than another) but also in terms of how much they differ (eg: one binary produces 10 more artefacts than another). Panik (2005) identifies this type of property as being *interval* data.

Under this scheme differences on a scale are meaningful and can be compared to other differences on the same scale. However, Panik points out that the zero point on the scale is deemed arbitrary; hence ratios of interval scale values are meaningless. To illustrate his point, Panik describes the issue of comparing the skill level of two golf players. This example has been adapted below to compare two software tools.

Consider the number of ports opened by a malware binary, as observed by two tools tested using the MATEF. Suppose the malware is programmed to open 3 ports on each execution and that the Oracle states that 2 ports are opened when the malware is executed. When tested, Tool A reports that 4 ports were opened, whilst Tool B states that 5 were opened. To answer the question of how good is Tool A at detecting open ports compared to Tool B depends on the point of reference or zero point. If the zero point is taken to be the Oracle, then the absolute difference between what was expected and what was observed is 2 for Tool A and 3 for Tool B. It might be tempting then to suggest that Tool A is one and one-half times as good as Tool B at observing open ports.

Alternatively, if the zero point is taken to be the programmed number of ports to be opened, then the absolute difference between what was expected and what was observed is 1 for Tool A and 2 for Tool B. In this scenario, one might argue that Tool A is now twice as good as Tool B at observing open ports.

Hence, the action of observing the number of artefacts produced by malware operates at the interval level of measurement. This conclusion has implications for the statistical tests that can be applied to this data (see section 4.6.2) and identifies the valid operations that can be performed upon it, see Table 4-8.

The comparison of a pair of observed values (one for each execution time, for Hypothesis 1) would only provide a measure of how similar the distribution of observations are at different lengths of execution. By also including the expected number of artefacts for each data point and calculating the absolute difference of the corresponding observed value from this, it becomes possible to gain a measure of the error in each observation. Note that 'error' in this context is defined as the difference between an estimated 'ground truth' (as provided by the Oracle) and the observed value. Plotting the frequency of these errors then produces a distribution of the absolute error of the observations between two execution times.

The error value is of no relevance to either Hypothesis 1 (Table 4-2) or Hypothesis 2 (Table 4-3). This is because both these hypotheses seek only to compare the quantity of artefacts. However, recording the magnitude of the error additionally provides a measure of how well the tool is performing against the Oracle. Thus each tool tested can be compared to both the approximated ground truth as well as other tools.

4.6.2 Deciding how to analyse it

The aim of any analysis to be performed is to address both hypotheses and ultimately the Research Question for this work. Both hypotheses are concerned with 'paired observations', where the subject (in this case a malware binary) is measured before and after a change of the independent variable.

Statistical tests that work with paired observations are generally divided into parametric and nonparametric tests. Parametric tests are applied to data that have a known (e.g.: *normal*) probability distribution, making it relatively easy to predict a future observation. Such data has fixed parameters, meaning that it is symmetrical about a central tendency and has a predictable spread of values.

Where the distribution of the data is not symmetrical, such as when the data is *skewed* to the left or the right, then nonparametric tests may be more appropriate.

A well known parametric statistical tests is the Student's t-Test. Field (2013, p. 165) describes the use of this test as being based on the assumption that the data to be analysed is normally distributed.

This assumption was not readily determinable *a priori* for the data in this research. Hence initial pilot studies were undertaken *post-hoc* to perform exploratory tests to address this assumption prior to performing any further statistical analysis.

4.6.3 Pilot studies to test for normality

To determine if the distribution of the data is normally distributed, the following *post-hoc* methodology was undertaken.

Methodology

To mitigate any anti-analysis technique in place (see section 4.1) the initial stage of the methodology sought to differentiate highly variable malware binaries from those that were less variable and hence more repeatable.

The approach taken follows that outlined in the section 4.5 (Experiment Design). In summary, the entire population in the malware library was examined by an online malware analysis platform, referred to in this research as the Oracle. All the malware binaries reported to exhibit network port behaviour were allocated to a group. From this group, 4,800 distinct malware binaries were chosen at random and allocated to a sub-group. The malware binaries in this sub-group then became the subjects of the test. This will be referred to as Dataset A.

The following procedure was then applied, first using the *Process Monitor* tool to record the observations (Pilot Study 1) and then using the *TCPVCon* tool to record the observations (Pilot Study 2).

Test procedure

Each malware binary in the sub-group was executed three times for the duration of ten seconds, recording the number of observed ports opened by the malware. Each binary was then again executed three times for the duration of one minute, again observing the opened ports. This resulted in two groups of three datasets, see Figure 4-4 and Table 4-9.

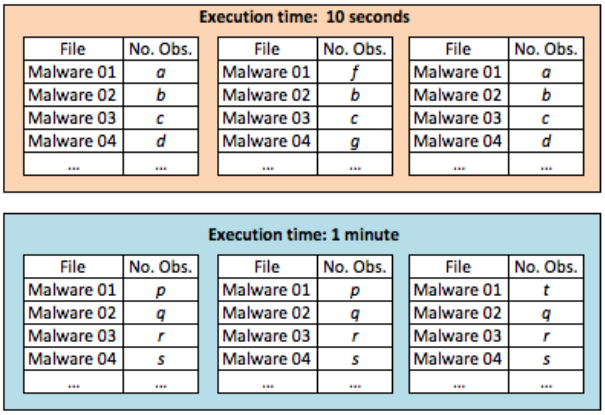

Figure 4-4 : Pilot study dataset structure

Pilot Study	Test Run ID	Dataset ID	Description	# of Malware Samples
1	114	A	Process Monitor, run for 1 minute	4091
1	126	A	Process Monitor, run for 1 minute	3726
1	127	A	Process Monitor, run for 1 minute	3794
1	147	A	Process Monitor, run for 10 seconds	3924
1	148	A	Process Monitor, run for 10 seconds	3743
1	149	A	Process Monitor, run for 10 seconds	3804
2	115	A	TCPVCon, run for 1 minute	3395
2	128	A	TCPVCon, run for 1 minute	3756
2	129	A	TCPVCon, run for 1 minute	3593
2	150	A	TCPVCon, run for 10 seconds	4046
2	151	A	TCPVCon, run for 10 seconds	3795
2	152	A	TCPVCon, run for 10 seconds	3958

Table 4-9 : Pilot studies - Initial datasets

Malware binaries that did not exhibit the same number of observations in all three datasets when executed for ten seconds were then filtered out; compare *Malware 01* and *Malware 04* in Figure 4-4 and Figure 4-5. Similarly, malware binaries that did not exhibit the same number of observations in all three datasets when executed for one minute were also filtered out; compare *Malware 01* in Figure 4-4 and Figure 4-5. Duplicate observations were then discarded to leave a single dataset of observations for each of the two execution times, see Figure 4-5 and Table 4-10.

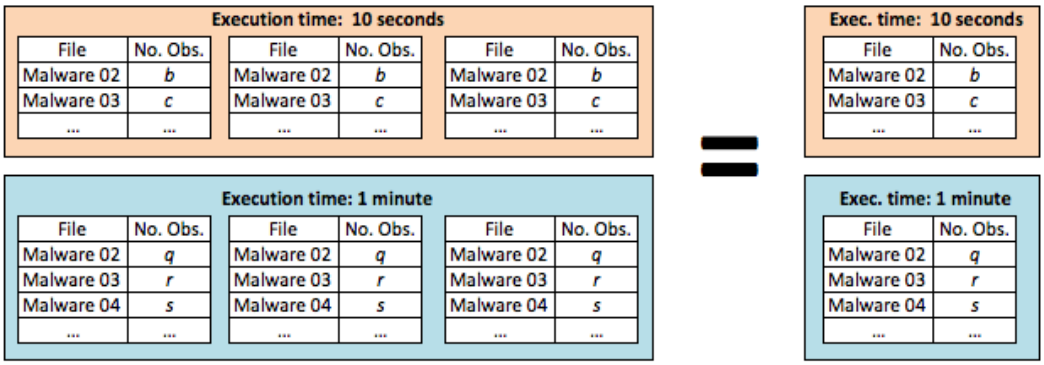

Figure 4-5 : Deduplicating repeatable observations

ID	Dataset Name	Description	# of Malware Samples
1	127_126_114	Process Monitor, repeatable values for 1 minute	2803
2	149_148_147	Process Monitor, repeatable values for 10 seconds	416
3	129_128_115	TCPVCon, repeatable values for 1 minute	1259
4	152_151_150	TCPVCon, repeatable values for 10 seconds	632

Table 4-10 : Pilot Studies - Repeatable datasets

With the variability in the malware minimised, a *post-hoc* assessment was performed to determine if the assumption of normality is violated or not.

To do this, each pair of datasets in Table 4-10 (grouped by tool) was combined into a single dataset of 'paired observations'. This meant that all of the malware binaries with observations in BOTH datasets (i.e.: binaries with observations for 1 minute and 10 seconds) were copied to a new, combined dataset for each tool, see Figure 4-6.

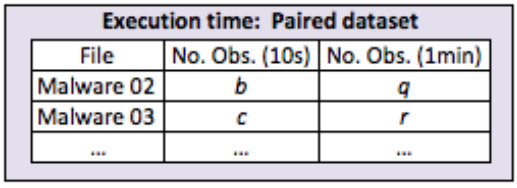

Figure 4-6 : Paired dataset for a tool

At this point each tool (Process Monitor and TCPVCon) each has a single dataset of paired values, containing observations from execution times of 10 seconds and one minute, see Table 4-11.

ID	Dataset Name	Description	# of Malware Samples
1	ProcessMon_1min10sec	Process Monitor, paired 1 min and 10 sec	333
2	TCPVCon_1min10sec	TCPVCon, paired 1 min and 10 sec	274

Table 4-11 : Pilot studies - Tool paired observations datasets

Checking for Normally distributed data

An absolute error value was calculated from the expected and observed number of artefacts in each dataset. A bi-modality was identified in both of the datasets listed in Table 4-11, see Figure 4-7 and Figure 4-8 of Pilot Study 1; also see Figure 4-9 and Figure 4-10 of Pilot Study 2. The datasets were therefore each split into two groups, above and below the threshold of this bi-modality (Absolute Error of 50). Each of the resulting subsets of data was then tested for Normality using the standard **Kolmogorov-Smirnoff** (K-S) and **Shapiro-Wilk** (S-W) tests.

Study	**Pilot Study 1 Process Monitor**		**Pilot Study 2 TCPVCon**	
Dataset	**K-S**	**S-W**	**K-S**	**S-W**
Execution time: 1 min Absolute Error >=50	0.000	0.000	0.000	0.000
Execution time: 1 min Absolute Error <50	0.065	**0.015**	0.200	**0.024**
Execution time: 10 sec Absolute Error >=50	0.000	0.000	0.000	0.000
Execution time: 10 sec Absolute Error <50	0.200	**0.037**	0.173	**0.027**

Table 4-12 : Pilot studies - Normality test results showing levels of significance

The K-S and S-W tests work by comparing the distribution against a normalised distribution. Hence, as Field (2013, p. 187) states, a significance value of less than 0.05 indicates a statistically significant deviation from a Normal distribution. Therefore the distribution producing such as result can be considered to not follow a Normal distribution.

Field goes on to argue that the S-W test has more power to detect a difference from normality than the K-S test. This may account for the difference of significance values in Table 4-12.

Referring to Table 4-12 it can be seen that for both the 1 minute and 10 second distributions where the absolute error is below 50 (shown in **bold** in the table) the resulting distribution does not follow a Normal distribution, according to the more powerful S-W test. This phenomenon occurs for both tools (Pilot Study 1 and 2). This would indicate that for on-going analysis parametric tests are not suitable and therefore nonparametric tests should be used instead.

Selecting an appropriate statistical test

Having established the need to use a nonparametric test, there remained the decision on which test to apply to compare these distributions. Recall the data is comprised of dependent observations, i.e.: numbers of artefacts observed for a given subject at different execution times. Hence, consideration needed only to be given to nonparametric tests that operate on dependent (as opposed to independent) samples.

The next question to consider was how many distributions were to be compared. Comparing multiple distributions together avoids *familywise* errors, whereby multiple Type I errors are introduced as a result of combining the results of multiple independent tests (Field, 2013, p. 68). However, tests that combine multiple distributions simply report that the distributions either are or are not the same. In other words, such tests do not identify which distributions are different. Hence, although such a test would address H_1 (*Does changing the execution time affect how many artefacts are observed?*), it would not highlight at what execution time this happens. Furthermore, the multi-modal nature of the data (see above) highlighted a difference in the parametric nature of the data above and below the absolute error of 50. This change in the nature of the data would be lost if multiple distributions were used collectively instead.

In addition, a test comparing multiple distributions would not address H_2 (*Which tool observes more artefacts?*) as this hypothesis must compare distributions taken under the same conditions, i.e.: a single execution time.

In conclusion therefore, a nonparametric test that compares two distributions of interval-based dependent variables is required.

The *Wilcoxon Signed Rank* test (Sheskin, 2011, p. 809) was selected to perform the analysis. This is because it is an established and appropriate statistical test for comparing distributions containing paired observations that are not normally distributed; or where one or more of the assumptions for the equivalent *t test* are saliently violated. The test accepts either ordinal or interval data.

An alternative considered was the *binomial sign* test (Sheskin, 2011, p. 823) as this is similar to the *Wilcoxon Signed Rank*. However, it only operates on the direction of differences and so ignores the magnitude of the differences. Hence this test has less power, meaning it is less capable of detecting changes compared to the *Wilcoxon Signed Rank* test.

Another alternative to the *Wilcoxon Signed Rank* test commonly considered is the *McNemar* test (Sheskin, 2011, p. 835). This test was discounted as it only supports nominal data and so would only indicate if one tool observed greater or fewer artefacts than another. It would not report by how much one tool was better (or worse) than another. Furthermore, as with the *binomial sign* test, this test has less power when compared to the *Wilcoxon Signed Rank* test, meaning it is less capable of detecting changes.

Pilot study 1 – Data produced by Process Monitor

A dataset of observations obtained using the tool Process Monitor v3.01 was selected. This data contains paired observations for malware binaries executed for both 10 seconds and 1 minute. The dataset contains 333 paired observations for malware binaries (N=333).

An initial analysis of the frequency distribution was produced for each condition (execution time) to explore the distribution of the data in each case, see

Figure 4-7 and

Figure 4-8.

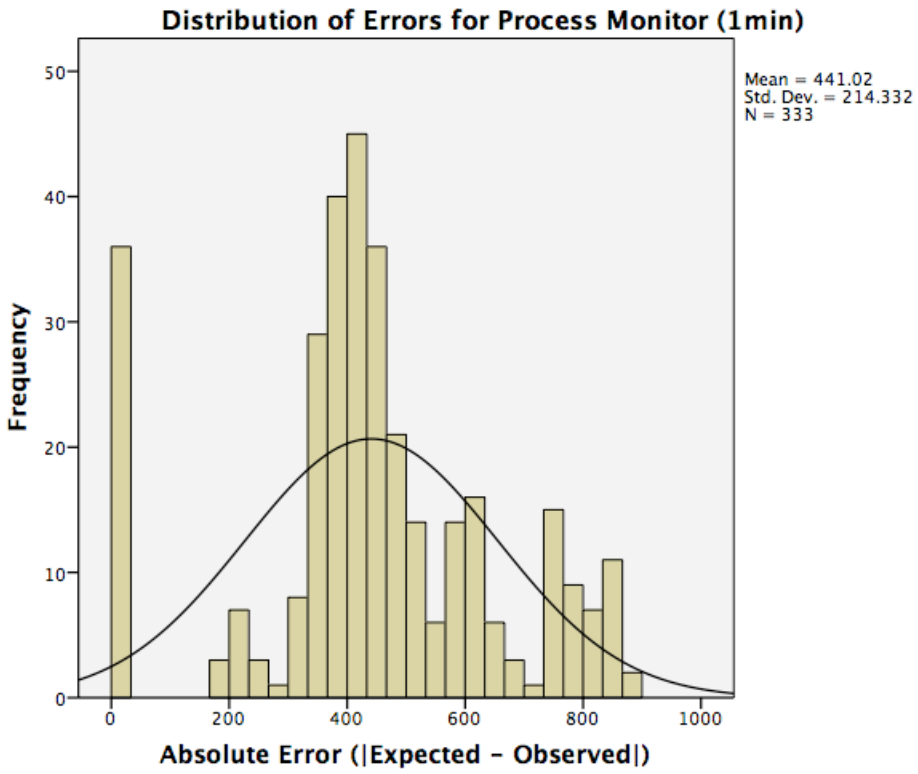

Figure 4-7 : Initial Frequency Distribution (Process Monitor - 1 min)

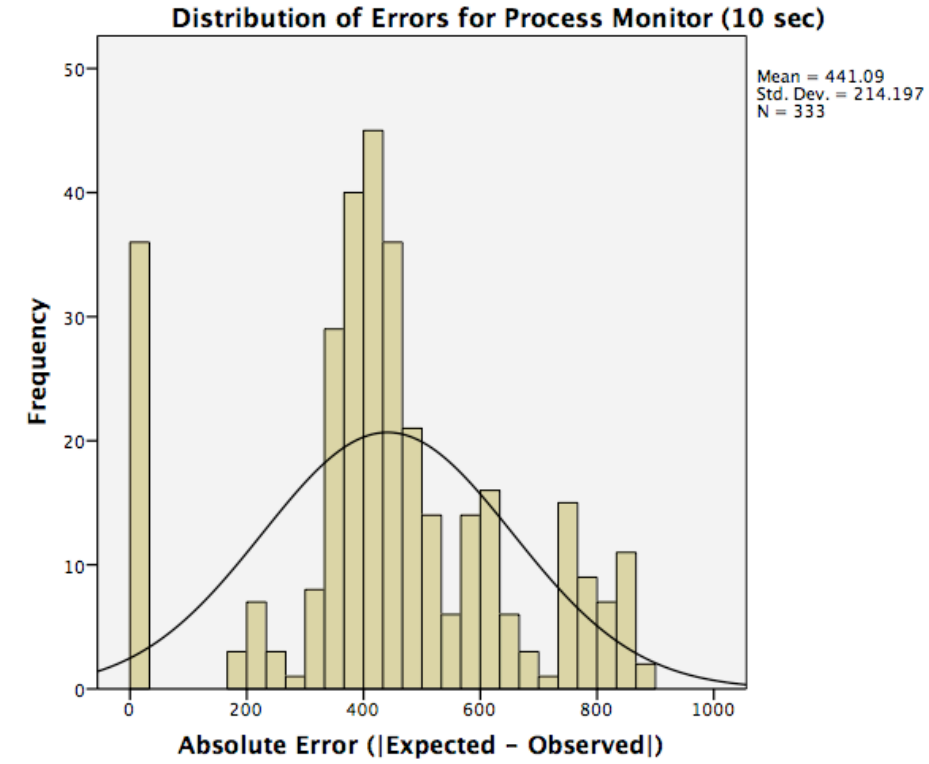

Figure 4-8 : Initial Frequency Distribution (Process Monitor - 10 sec)

In both cases there is a clear bi-modality in the data for absolute errors of approximately 50 or less. This was confirmed by examining the frequency table produced for each of the two distributions.

The dataset was therefore split into two groups, subjects with an absolute error less than 50 and those with an absolute error of 50 or more. Each of the four resulting subsets of data were then tested for Normality using the standard **Kolmogorov-Smirnoff** (K-S) and **Shapiro-Wilk** (S-W) tests. See the discussion on Checking for Normally distributed data, above.

Pilot Study 2 – Examining data produced by TCPVCon

A dataset of observations obtained using the tool TCPVCon v3.01 was selected. This data contains paired observations for malware binaries executed for both 10 seconds and 1 minute. The dataset contains 274 paired observations for malware binaries (N=274).

An initial analysis of the frequency distribution was produced for each condition (execution time) to explore the distribution of the data in each case, see Figure 4-9 and Figure 4-10.

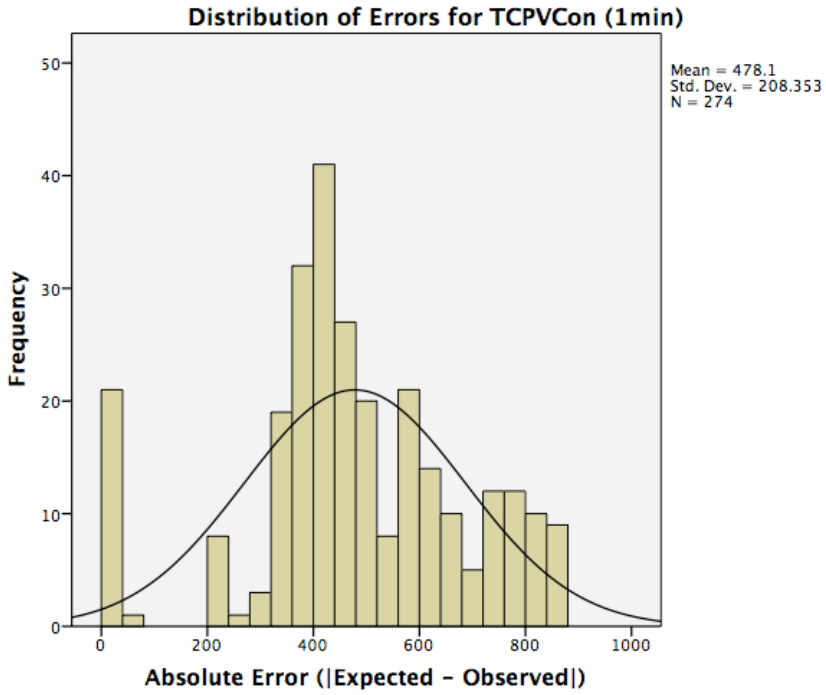

Figure 4-9 : Initial Frequency Distribution (TCPVCon - 1 min)

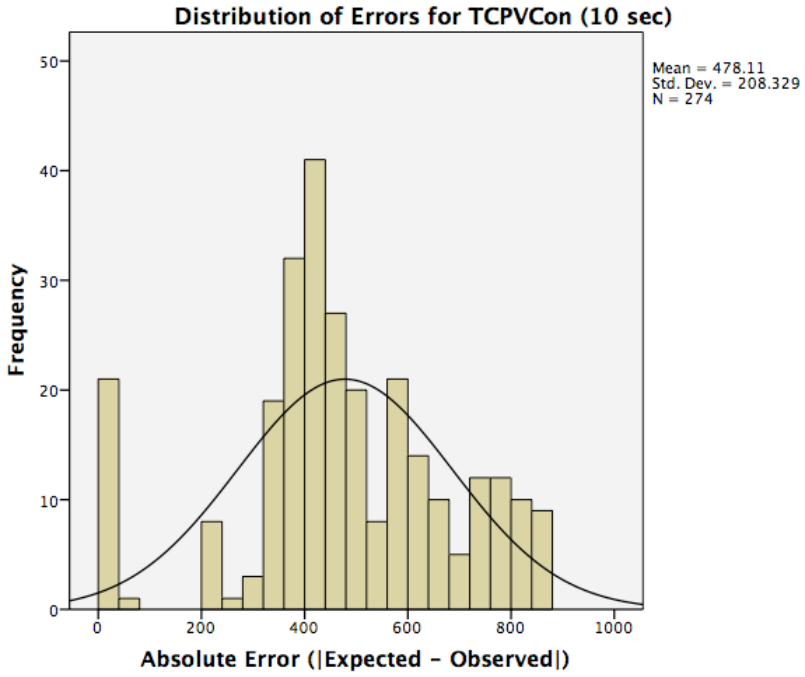

Figure 4-10 : Initial Frequency Distribution (TCPVCon - 10 sec)

As before, both cases demonstrate a clear bi-modality in the data for absolute errors of approximately 50 or less. This was confirmed by examining the frequency table produced for each of the two distributions.

As with the Process Monitor data, the dataset was split into two groups, subjects with an absolute error less than 50 and those with an absolute error of 50 or more. Each of the four resulting subsets of data were then tested for Normality using the standard **Kolmogorov-Smirnoff** (K-S) and **Shapiro-Wilk** (S-W) tests. As before, see the discussion on Checking for Normally distributed data, above.

4.7 Chapter Summary

This chapter has identified the aims of the MATEF design by linking back to the previously identified requirements. With the design aims established, the main components of the framework were conceptualised to meet each of the design aims. The main components identified include a source for real-world malware to provide a greater element of validity to the results as possible. Also included is a component that acts as a source of *ground truth* about the malware is referred to as the *Oracle*. This Oracle would determine the expected behaviour of a given malware binary.

Once ingested, a malware binary is passed to two further components, namely a library to house the malware binaries and a database to store details concerning each binary.

Management scripts form another significant component of the MATEF design and are responsible for managing the database and library along with the tests themselves, producing standardised log files ready for analysis.

The component that forms the test environment is a collection of virtualised operating systems running in parallel. The choice of operating system is such that it provides a fertile an environment as possible for malware to operate and hence provide the 'best case scenario' for tools under analysis.

The final component is the analysis part of the MATEF. The design of this component is informed by the aims of the framework as a whole (see section 4.1). Hence the choice of metrics to monitor, the design of the test runs performed and the statistical analysis performed are all determined by the objectives. As a result, two pairs of hypotheses have been identified which ask two fundamental questions: "Does changing the execution time affect how many artefacts are observed" (see Table 4-2) and "Which of two tools observes more artefacts under the same conditions" (see Table 4-3).

The next chapter applies the methodology outlined and presents a more detailed analysis of the results relating to these hypotheses.

Chapter 5 Results and analysis

In the previous chapter the MATEF was designed from the requirements and implemented to meet the identified aims using resources available to the research project. To demonstrate its utility, a series of experiments were designed together with an associated analysis strategy. This chapter presents the results and subsequent analysis of these experiments to demonstrate the analysis and conclusions that can be drawn from data collated using the MATEF. It should be noted that these experiments and subsequent analysis are only examples of what can be achieved using the MATEF. The results of these experiments show how the choice of tool can determine the optimum execution time used to monitor malware.

The chapter opens with some worked examples on how the analysis was performed (5.1). After this a summary of the results is presented (5.2), which is then followed by a discussion (5.3). Conclusions are then drawn (5.4) and finally the chapter is summarised (5.5).

5.1 Worked examples of analysis

With the *Wilcoxon Signed Rank* test selected in the previous chapter, what follows are two worked examples to demonstrate how the *Wilcoxon Signed Rank* test was applied to the output of two different tools. The data for these worked examples is taken from the Pilot study (see section 4.6.3).

5.1.1 Worked example of analysis for Process Monitor

The paired observation data for the *Process Monitor* tool (Dataset ID 1 in Table 4-11) was loaded in SPSS (IBM, 2016), see Table 5-1 for a sample of the data. Note the hash values here have been shortened for brevity.

MD5 Hash	Expected	Observed (1min)	Abs. Error (1min)	Observed (10sec)	Abs. Error (10sec)
fe40…271d	751	0	751	0	751
05b4…3ae1	3	3	0	1	2
7ccf…e04e	9	0	9	0	9
183e…b7a4	7	0	7	0	7

Table 5-1 : Sample of data from dataset *ProcessMon_1min10sec*

Recall from section 4.2.8 that **Hypothesis 1** (H_1) is stated as:

$H_{1.0}$ Changing the execution time of malware has no significant impact on the number of malware artefacts observed by a given tool.

$H_{1.1}$ Changing the execution time of malware has a significant impact on the number of malware artefacts observed by a given tool.

The two absolute error values from the ProcessMon_1min10sec dataset were supplied to a Wilcoxon Signed Rank test. This produced a rejection of the Null Hypothesis ($H_{1.0}$):

Hypothesis Test Summary

	Null Hypothesis	Test	Sig.	Decision
1	The median of differences between ProcessMon_1min and ProcessMon_10sec equals 0.	Related-Samples Wilcoxon Signed Rank Test	.011	Reject the null hypothesis.

Asymptotic significances are displayed. The significance level is .05.

Total N	333
Test Statistic	28.000
Standard Error	5.534
Standardized Test Statistic	2.530
Asymptotic Sig. (2-sided test)	.011

Figure 5-1 : Worked example 1 (Process Monitor)

The effect size (r) is given by:

$$r = \frac{z}{\sqrt{N}} = \frac{2.530}{\sqrt{333}} = 0.1386$$

Equation 5-1 : Effect size for Process Monitor (Pilot study)

This means that for the *Process Monitor* tool, the differences between the expected and observed number of ports opened during a 1 minute execution time (Median=427) were significantly different to the differences between the expected and observed number of ports opened during a 10 second execution time (Median=427), $T = 28$, $p = 0.011$, $r = 0.1386$.

5.1.2 Worked example of analysis for TCPVCon

The paired observation data for the *TCPVCon* tool (Dataset ID 2 in Table 4-11) was loaded in SPSS, see Table 5-2 for a sample of the data. As before, the hash values here have been shortened for brevity.

MD5 Hash	Expected	Observed (1min)	Abs. Error (1min)	Observed (10sec)	Abs. Error (10sec)
ff18…e59a	483	0	483	0	483
4832…3537	10	2	8	0	10
4c43…af0c	8	7	1	7	1
5313…688e	1	1	0	1	0

Table 5-2 : Sample of data from dataset *TCPVCon_1min10sec*

As before, the two absolute error values from the TCPVCon_1min10sec dataset were supplied to a Wilcoxon Signed Rank test. This produced a failure to reject the Null Hypothesis ($H_{1.0}$):

Hypothesis Test Summary

	Null Hypothesis	Test	Sig.	Decision
1	The median of differences between TCPVCon_1min and TCPVCon_10sec equals 0.	Related-Samples Wilcoxon Signed Rank Test	.180	Retain the null hypothesis.

Asymptotic significances are displayed. The significance level is .05.

Total N	274
Test Statistic	3.000
Standard Error	1.118
Standardized Test Statistic	1.342
Asymptotic Sig. (2-sided test)	.180

Figure 5-2 : Worked example 2 (TCPVCon)

Calculating the effect size (r) gives:

$$r = \frac{z}{\sqrt{N}} = \frac{1.342}{\sqrt{274}} = 0.0811$$

Equation 5-2 : Effect size for TCPVCon (Pilot study)

This means that for *TCPVCon* the differences between the expected and observed number of ports opened during a 1 minute execution time (Median=456.5) were not significantly different to the differences between the expected and observed number of ports opened during a 10 minute execution time (Median=456.5), $T = 3.0$, $p = 0.180$, $r = 0.0811$.

Having described the analysis process with these two worked examples, what follows is a more comprehensive summary of the results.

5.2 Experimental results

Prior to presenting a summary of these results here it is worth recapping the hypotheses (H_1 and H_2) presented respectively in Table 4-2 and Table 4-3 within section 4.2.8:

$H_{1.0}$	Changing the execution time of malware has no significant impact on the number of malware artefacts observed by a given tool.
$H_{1.1}$	Changing the execution time of malware has a significant impact on the number of malware artefacts observed by a given tool.

Hypothesis 1

$H_{2.0}$	There is no significant difference on the number of malware artefacts observed by Tool A when compared to Tool B, under the same conditions.
$H_{2.1}$	Tool A is able to detect a significantly greater number of artefacts when compared to Tool B, under the same conditions.
$H_{2.2}$	Tool B is able to detect a significantly greater number of artefacts when compared to Tool A, under the same conditions.

Hypothesis 2

Recall the malware dataset has been partitioned into binaries identified as repeatable and non-repeatable (see section 4.2.8). These hypotheses are only applicable for repeatable malware.

The experiment results follow and are summarised in Table 5-3 below:

Experiment	Test	Description	Result
1	1.1	Comparing 1 minute to 5 minute execution times of Process Monitor	Retain $H_{1.0}$
	1.2	Comparing 1 minute to 10 minute execution times of Process Monitor	Retain $H_{1.0}$
	1.3	Comparing 1 minute to 10 seconds execution times of Process Monitor	Reject $H_{1.0}$ Propose $H_{1.1}$
2	2.1	Comparing 1 minute to 5 minute execution times of TCPVCon	Retain $H_{1.0}$
	2.2	Comparing 1 minute to 10 minute execution times of TCPVCon	Retain $H_{1.0}$
	2.3	Comparing 1 minute to 10 seconds execution times of TCPVCon	Retain $H_{1.0}$
3	3.1	Process Monitor vs TCPVCon, run for 10 seconds	Retain $H_{2.0}$
	3.2	Process Monitor vs TCPVCon, run for 1 minute	Retain $H_{2.0}$
	3.3	Process Monitor vs TCPVCon, run for 5 minutes	Retain $H_{2.0}$
	3.4	Process Monitor vs TCPVCon, run for 10 minutes	Retain $H_{2.0}$

Table 5-3 : Results relating to Hypothesis 1 and Hypothesis 2

5.2.1 Experiment 1 - Comparing Process Monitor at different execution times

To address Hypothesis 1 in Table 4-2 and determine the impact of execution time (if any) on the number of artefacts observed by the Process Monitor tool, the distributions were analysed in pairs, 1 minute vs 5 minutes and 1 minute vs 10 minutes. The Wilcoxon signed-rank test was applied to each pair of samples and produced the following results:

Test 1.1 – Comparing 1 minute to 5 minutes of execution time

The results for the 1 minute vs 5 minutes execution time are as follows:

Hypothesis Test Summary

	Null Hypothesis	Test	Sig.	Decision
1	The median of differences between ProcessMon_1min and ProcessMon_5min equals 0.	Related-Samples Wilcoxon Signed Rank Test	.068	Retain the null hypothesis.

Asymptotic significances are displayed. The significance level is .05.

Figure 5-3 : Test 1.1 Hypothesis Test Summary

Total N	829
Test Statistic	10.000
Standard Error	2.739
Standardized Test Statistic	1.826
Asymptotic Sig. (2-sided test)	.068

Figure 5-4 : Test 1.1 Results summary

The effect size (r) is given by:

$$r = \frac{z}{\sqrt{N}}$$

Equation 5-3 : Calculating the effect size

where z is the Standardised Test Statistic and N is the number of observations.

Thus, for Test 1.1, the effect size (r) is

$$r = \frac{z}{\sqrt{N}} = \frac{1.826}{\sqrt{829}} = 0.0634$$

Equation 5-4 : Effect size for Test 1.1

These results mean that for Process Monitor the differences between the expected and observed number of ports opened during a 1 minute execution time (Median=390) were identical to the differences between the expected and observed number of ports opened during a 5 minute execution time (Median=390), $T = 10$, $p = 0.068$, $r = 0.0634$.

Test 1.2 – Comparing 1 minute to 10 minutes of execution time

The results for the 1 minute vs 10 minutes execution time are as follows:

Hypothesis Test Summary

	Null Hypothesis	Test	Sig.	Decision
1	The median of differences between ProcessMon_1min and ProcessMon_10min equals 0.	Related-Samples Wilcoxon Signed Rank Test	.175	Retain the null hypothesis.

Asymptotic significances are displayed. The significance level is .05.

Figure 5-5 : Test 1.2 Hypothesis Test Summary

Total N	1,056
Test Statistic	22.000
Standard Error	5.895
Standardized Test Statistic	1.357
Asymptotic Sig. (2-sided test)	.175

Figure 5-6 : Test 1.2 Results Summary

Applying Equation 5-3, for Test 1.2 *mutatis mutandis*, the effect size (*r*) is

$$r = \frac{z}{\sqrt{N}} = \frac{1.357}{\sqrt{1056}} = 0.0418$$

Equation 5-5 : Effect size for Test 1.2

These results mean that for Process Monitor the differences between the expected and observed number of ports opened during a 1 minute execution time (Median=386.5) were identical to the differences between the expected and observed number of ports opened during a 10 minute execution time (Median=386.5), $T = 22$, $p = 0.175$, $r = 0.0418$.

Observation

Although both Test 1.1 and 1.2 produced a result where the null hypothesis is retained, there is a drop in the significance from 0.175 (Test 1.2) to 0.068 (Test 1.1), which is not far above the threshold of 0.05.

In light of this observation, Process Monitor was again used to gather observations on the same malware, but this time executing the malware and tool for a period of 10 seconds. The results are presented in Test 1.3.

Test 1.3 – Comparing 1 minute to 10 seconds of execution time

The two absolute error values from the ProcessMon_1min10sec dataset were supplied to a Wilcoxon Signed Rank test. This produced a rejection of the Null Hypothesis ($H_{1.0}$):

Hypothesis Test Summary

	Null Hypothesis	Test	Sig.	Decision
1	The median of differences between ProcessMon_1min and ProcessMon_10sec equals 0.	Related-Samples Wilcoxon Signed Rank Test	.011	Reject the null hypothesis.

Asymptotic significances are displayed. The significance level is .05.

Figure 5-7 : Test 1.3 Hypothesis Test Summary

Total N	333
Test Statistic	28.000
Standard Error	5.534
Standardized Test Statistic	2.530
Asymptotic Sig. (2-sided test)	.011

Figure 5-8 : Test 1.3 Results Summary

Applying Equation 5-3, for Test 1.3 *mutatis mutandis*, the effect size (r) is

$$r = \frac{z}{\sqrt{N}} = \frac{2.530}{\sqrt{333}} = 0.1386$$

Equation 5-6 : Effect size for Test 1.3

These results mean that for Process Monitor the differences between the expected and observed number of ports opened during a 1 minute execution time (Median=427) were significantly different to the differences between the expected and observed number of ports opened during a 10 second execution time (Median=427), $T = 28$, $p = 0.011$, $r = 0.1386$.

5.2.2 Experiment 2 - Comparing TCPVCon at different execution times

Again, to address Hypothesis 1 in Table 4-2 and determine the impact of execution time (if any) on the number of artefacts observed by a different tool (TCPVCon), the distributions were analysed in pairs, 1 minute vs 5 minutes and 1 minute vs 10 minutes. The Wilcoxon signed-rank test was again applied to each pair of samples and produced the following results:

Test 2.1 – Comparing 1 minute to 5 minutes of execution time

The results for the 1 minute vs 5 minutes execution time are as follows:

Hypothesis Test Summary

	Null Hypothesis	Test	Sig.	Decision
1	The median of differences between TCPVCon_1min and TCPVCon_5min equals 0.	Related-Samples Wilcoxon Signed Rank Test	.655	Retain the null hypothesis.

Asymptotic significances are displayed. The significance level is .05.

Figure 5-9 : Test 2.1 Hypothesis Test Summary

Total N	675
Test Statistic	1.000
Standard Error	1.118
Standardized Test Statistic	-.447
Asymptotic Sig. (2-sided test)	.655

Figure 5-10 : Test 2.1 Results Summary

Applying Equation 5-3, for Test 2.1 *mutatis mutandis*, the effect size (r) is

$$r = \frac{z}{\sqrt{N}} = \frac{-0.447}{\sqrt{675}} = -0.0172$$

Equation 5-7 : Effect size for Test 2.1

These results mean that for TCPVCon the differences between the expected and observed number of ports opened during a 1 minute execution time (Median=421) were significantly different to the differences between the expected and observed number of ports opened during a 5 minute execution time (Median=421), $T = 1.0$, $p = 0.655$, $r = -0.0172$.

Test 2.2 – Comparing 1 minute to 10 minutes of execution time

The results for the 1 minute vs 10 minutes execution time are as follows:

Hypothesis Test Summary

	Null Hypothesis	Test	Sig.	Decision
1	The median of differences between TCPVCon_1min and TCPVCon_10min equals 0.	Related-Samples Wilcoxon Signed Rank Test	1.000	Retain the null hypothesis.

Asymptotic significances are displayed. The significance level is .05.

Figure 5-11 : Test 2.2 Hypothesis Test Summary

Total N	569
Test Statistic	.000
Standard Error	.000
Standardized Test Statistic	NaN
Asymptotic Sig. (2-sided test)	1.000

Figure 5-12 : Test 2.2 Results Summary

Because the Standard Error (SE) is zero, the Test Statistics (z) cannot be calculated and thus the effect size cannot be determined. Furthermore, an SE value of zero indicates the median of the differences between the two distributions (the 1 minute and 10 minute execution times) is also zero, i.e.: there is no change between the two distributions.

Test 2.3 – Comparing 1 minute to 10 seconds of execution time

The results for the 1 minute vs 10 seconds execution time are as follows:

Hypothesis Test Summary

	Null Hypothesis	Test	Sig.	Decision
1	The median of differences between TCPVCon_1min and TCPVCon_10sec equals 0.	Related-Samples Wilcoxon Signed Rank Test	.180	Retain the null hypothesis.

Asymptotic significances are displayed. The significance level is .05.

Figure 5-13 : Test 2.3 Hypothesis Test Summary

Total N	274
Test Statistic	3.000
Standard Error	1.118
Standardized Test Statistic	1.342
Asymptotic Sig. (2-sided test)	.180

Figure 5-14 : Test 2.3 Results Summary

Applying Equation 5-3, for Test 2.3 *mutatis mutandis*, the effect size (r) is

$$r = \frac{z}{\sqrt{N}} = \frac{1.342}{\sqrt{274}} = 0.0811$$

Equation 5-8 : Effect size for Test 2.3

These results mean that for TCPVCon the differences between the expected and observed number of ports opened during a 1 minute execution time (Median=456.5) were not significantly different to the differences between the expected and observed number of ports opened during a 10 minute execution time (Median=456.5), $T = 3.0$, $p = 0.180$, $r = 0.0811$

5.2.3 Experiment 3 - Comparing Process Monitor and TCPVCon

To address Hypothesis 2 in Table 4-3 and inform the practitioner's choice of tool, distributions for the same execution time from each tool were analysed for the execution times of 10 seconds, 1 minute, 5 minutes and 10 minutes. The Wilcoxon signed-rank test was again applied to each pair of samples and produced the following results:

Test 3.1 – Comparing Process Monitor and TCPVCon for 10 seconds of execution time

The results for the 10 seconds of execution time are as follows:

Hypothesis Test Summary

	Null Hypothesis	Test	Sig.	Decision
1	The median of differences between ProcessMon_10sec and TCPVCon_10sec equals 0.	Related-Samples Wilcoxon Signed Rank Test	1.000	Retain the null hypothesis.

Asymptotic significances are displayed. The significance level is .05.

Figure 5-15 : Test 3.1 Hypothesis Test Summary

Total N	125
Test Statistic	.000
Standard Error	.000
Standardized Test Statistic	NaN
Asymptotic Sig. (2-sided test)	1.000

Figure 5-16 : Test 3.1 Results Summary

Because the Standard Error (SE) is zero, the Test Statistics (z) cannot be calculated and thus the effect size cannot be determined. Furthermore, an SE value of zero indicates the median of the differences between the two distributions (Process Monitor and TCPVCon run for 10 seconds of execution times) is also zero, i.e.: there is no change between the two distributions.

Test 3.2 – Comparing Process Monitor and TCPVCon for 1 minute of execution time

The results for the 1 minute of execution time are as follows. Note in the Summary below, the field *AbsDiff_127* refers to the absolute differences observed by Process monitor and field *AbsDiff_129* refers to the absolute differences observed by TCPVCon.

Hypothesis Test Summary

	Null Hypothesis	Test	Sig.	Decision
1	The median of differences between AbsDiff_127 and AbsDiff_129 equals 0.	Related-Samples Wilcoxon Signed Rank Test	.056	Retain the null hypothesis.

Asymptotic significances are displayed. The significance level is .05.

Figure 5-17 : Test 3.2 Hypothesis Test Summary

Total N	994
Test Statistic	63.500
Standard Error	27.253
Standardized Test Statistic	-1.908
Asymptotic Sig. (2-sided test)	.056

Figure 5-18 : Test 3.2 Results Summary

Applying Equation 5-3, for Test 3.2 *mutatis mutandis*, the effect size (r) is

$$r = \frac{z}{\sqrt{N}} = \frac{-1.908}{\sqrt{994}} = -0.0605$$

Equation 5-9 : Effect size for Test 3.2

These results mean that when comparing Process Monitor to TCPVCon, the differences between the expected and observed number of ports opened during a 1 minute execution time were not significantly different to each other, $T = 63.5$, $p = 0.056$, $r = -0.0605$. Note however, the significance value (p) is close to being significant, i.e.: < 0.05 in value.

Test 3.3 – Comparing Process Monitor and TCPVCon for 5 minutes of execution time

The results for the 5 minutes of execution time are as follows. Note in the Summary below, the field *AbsDiff_135* refers to the absolute differences observed by Process monitor and field *AbsDiff_131* refers to the absolute differences observed by TCPVCon.

Hypothesis Test Summary

	Null Hypothesis	Test	Sig.	Decision
1	The median of differences between AbsDiff_131 and AbsDiff_135 equals 0.	Related-Samples Wilcoxon Signed Rank Test	.157	Retain the null hypothesis.

Asymptotic significances are displayed. The significance level is .05.

Figure 5-19 : Test 3.3 Hypothesis Test Summary

Total N	496
Test Statistic	3.000
Standard Error	1.061
Standardized Test Statistic	1.414
Asymptotic Sig. (2-sided test)	.157

Figure 5-20 : Test 3.3 Results Summary

Applying Equation 5-3, for Test 3.3 *mutatis mutandis*, the effect size (r) is

$$r = \frac{z}{\sqrt{N}} = \frac{1.414}{\sqrt{496}} = 0.0635$$

Equation 5-10 : Effect size for Test 3.3

These results mean that when comparing Process Monitor to TCPVCon, the differences between the expected and observed number of ports opened during a 5 minute execution time were not significantly different to each other, $T = 3.0$, $p = 0.157$, $r = 0.0635$

Test 3.4 – Comparing Process Monitor and TCPVCon for 10 minutes of execution time

The results for the 10 minutes of execution time are as follows. Note in the Summary below, the field *AbsDiff_137* refers to the absolute differences observed by Process monitor and field *AbsDiff_133* refers to the absolute differences observed by TCPVCon.

Hypothesis Test Summary

	Null Hypothesis	Test	Sig.	Decision
1	The median of differences between AbsDiff_137 and AbsDiff_133 equals 0.	Related-Samples Wilcoxon Signed Rank Test	.317	Retain the null hypothesis.

Asymptotic significances are displayed. The significance level is .05.

Figure 5-21 : Test 3.4 Hypothesis Test Summary

Total N	554
Test Statistic	.000
Standard Error	.500
Standardized Test Statistic	-1.000
Asymptotic Sig. (2-sided test)	.317

Figure 5-22 : Test 3.4 Results Summary

Applying Equation 5-3, for Test 3.4 *mutatis mutandis*, the effect size (r) is

$$r = \frac{z}{\sqrt{N}} = \frac{-1.0}{\sqrt{554}} = -0.0425$$

Equation 5-11 : Effect size for Test 3.3

These results mean that when comparing Process Monitor to TCPVCon, the differences between the expected and observed number of ports opened during a 10 minute execution time were not significantly different to each other, $T = 0.0$, $p = 0.317$, $r = -0.0425$

5.3 Analysis and discussion

Experiment 1 comprised of three tests relating to Hypothesis 1 (see previous section), which sought to determine the impact of execution time (if any) on the number of artefacts observed by the *Process Monitor* tool. For the *Process Monitor* tool, the results indicated that execution time has no statistically significant effect on the differences between the expected and observed number of ports opened during execution time until the execution time is reduced below one minute to ten seconds. Hence the optimal execution time to observe ports opened by malware using *Process Monitor* is between 10 seconds and one minute. Furthermore, there is no perceived benefit in executing *Process Monitor* to observe the number of ports opened by malware for more than one minute.

This result contrasts with the *TCPVCon* tool (Experiment 2) whose results indicated that execution time has no statistically significant impact on the outcome under the same range of execution times. Hence, Hypothesis 1 cannot be generalised to all tools, as it is subject to the tool being used. However, this result demonstrates that the MATEF has provided an empirical methodology to compare the impact of execution time on different tools. The knowledge gleaned from such tests can be therefore be used by a practitioner to inform their choice of tool when conducting malware analysis. Furthermore, this result means that when a practitioner obtains a new tool, the MATEF can be used to specify parameters, such as how long the tool must be run for to obtain the optimal number of artefacts.

The results for Experiment 3 (Hypothesis 2) indicated that there is no statistical difference between using *Process Monitor* over *TCPVCon* as a tool to capture port related artefacts.

5.4 Conclusions

It has been demonstrated that the MATEF can provide a systematic approach to observing malware artefacts under different conditions with different tools. Several control variables were used during the tests conducted. These include the operating environment, the malware binaries, the tools tested and the variability due to random anti-forensic techniques. By controlling these variables it has been possible to isolate variations in the results reported by tools due to execution time. The results identify an optimal execution time for a tool used to study malware artefacts as well as comparing two such tools.

The results also indicate that changing the execution time of a tool used to monitor activity resulting from malware can, depending on the tool used, have an effect on number of artefacts observed.

The impact of execution time on the results obtained from tools used for malware forensics has not been studied previously. For example, the five-phase model proposed by Malin *et al.* (2008) makes no reference to execution time (see section 2.2). In the absence of any guidance or knowledge on this, practitioners would be selecting and running tools without any knowledge of the impact their choice of tool or execution time could have on their results.

Therefore, identifying an appropriate tool can reduce the time required to observe the effect of malware and hence contribute towards reducing the time required to undertake a malware forensic investigation.

5.5 Chapter summary

This chapter outlined the results obtained from applying the analysis strategy presented in the previous chapter and presented the overall results obtained from using the MATEF. It concluded that the choice of tool used could have an effect on the execution time used to monitor malware. Furthermore, by formulating additional hypotheses regarding different aspects of malware analysis tools, the MATEF can be used to provide a basis for trusting malware forensic analysis.

The next chapter will define the success criteria for the MATEF and apply this to evaluate the MATEF against a variety of criteria, such as the aims and requirements of the project.

Chapter 6 Evaluation of the MATEF

In the preceding chapters the need for trusted practice in the use of software tools used in malware forensics has been identified. The requirements to address this gap have been specified and a framework has been designed and implemented. In addition, the use of the framework has been demonstrated to evaluate certain aspects of the malware analysis method (i.e. the impact of changing the tools used and their execution times). It still remains to evaluate the framework itself both from a functional perspective against the requirements and a quality perspective in terms of performance and resource utilisation.

Therefore, this chapter evaluates the MATEF against the original aims, requirements and research question. The chapter opens by identifying the evaluation criteria (6.1) and defining what success looks like before moving on to evaluate the MATEF against the requirements and aims (6.2). Evaluations in terms of performance (6.3) and the Research Question (6.4) are also considered before providing a discussion on the limitations of the framework (6.5). Conclusions and further work are presented (6.6) prior to a summary of the chapter (6.7).

6.1 Evaluation criteria

Prior to evaluating the framework it is useful to consider what criteria can be used to evaluate the level of success attained by the MATEF. Hence, understanding what criteria should exist in a successful framework will be considered prior to examining each of these criteria in turn.

A starting position to evaluate the MATEF is to consider how well it has met the requirements of the framework (see section 3.3) and further, how well it has achieved the aims of the framework (as set out in section 4.1).

A further measure is to consider how well the MATEF has addressed the fundamental motivation for the research, expressed through the Research Question. In addition, an assessment on the performance of the framework can be applied in terms of the speed and resource utilisation. Finally, an exercise in identifying any areas of improvement in the design and implementation of the MATEF will be undertaken. Wherever possible, mitigations for these are presented. Each of these criteria is considered in the sections that follow.

6.2 Evaluate against framework requirements and aims

Requirements

The requirements for the MATEF are divided into external and internal requirements (see sections 3.3.1 and 3.3.2 respectively). The MATEF has been evaluated against these requirements, the conclusions of which are summarised in Table 6-1 and Table 6-2 respectively.

#	**Requirement**	**Met / Not met**	**Rationale for decision**
1	Handling of malware and what it may access should be controlled.	Met	Malware binaries are handled via scripts on an internal network, isolated from the rest of the University/outside world.
2	Output of tested tool must be admissible.	Not met	This is untested as the MATEF has not yet been applied to a live case.
3	Malware analysis tool output must be 'reliable'	Met	The results of the analysis performed on the MATEF output are based on established statistical techniques.
4	Novel methods must be validated	Not met	Given there is no ground truth, this is not easily achieved.
5	The theory/technique should be peer reviewed or published	Met	The MATEF design, implementation and results are published in this dissertation
6	Method should be a generally accepted	Not met	It is too early in the project's lifecycle for the MATEF to have been accepted by others yet.
7	Use a VM	Met	Virtual machines are used extensively in the MATEF
8	Network service provision	Met	Network services are provided through a open source simulated network services.
9	Use vulnerable environment	Met	The Windows XP operating system is used to provide a fertile environment for the malware.

Table 6-1 : External requirements evaluation

#	**Requirement**	**Met / Not met**	**Rationale for decision**
1	Black box testing approach	Met	Closed source software tools have been used throughout.
2	Malware lab requirements: VM Only approach	Met	Virtual machines are used extensively in the MATEF
3	Malware lab requirements: Single operating system	Met	A single operating system has been implemented throughout.
4	Malware lab requirements: Configure the OS to be malware friendly	Met	The Windows XP operating system is used to provide a fertile environment for the malware.
5	Accept real-world malware from any source	Met	The MATEF can accept malware from any source as it simply requires the binary file(s) to be placed in a specified folder
6	Storing & handling malware: Avoid cross contamination	Met	Malware files are stored in encrypted password protected ZIP files. Prior to running a test a folder accessible to the VM is cleared and the malware file is decrypted an dcopied into this folder.
7	Storing & handling malware: Extract via automation	Met	Malware submissions to the Oracle for analysis and the decryption process referred to above are automated via scripts.
8	Storing & handling malware: Restrict access to malware	Met	Malware is stored in a folder with restricted permissions. Furthermore, as stated above, each malware binary is encrypted in a password protected ZIP file.
9	Metrics: Determine the expected quantity of artefacts from an independent source	Met	Each malware binary is submitted to the Oracle for analysis. This analysis provides (amongst other things) the expected number of artefacts.

Table 6-2 : Internal requirements evaluation

Commencing with the external requirements summarised in Table 6-1, six of the nine external requirements were met. These were achieved largely through the design of the framework. For example, the handling requirement (Requirement 1) is satisfied through the automated handling via a script (minimising the effects of human error) and the network configuration. Other requirements met by the design include the use of VMs (Requirement 7), network services provision (Requirement 8) and a fertile and vulnerable environment for malware (Requirement 9).

The design of the framework also contributed to the integration of the scientific method, satisfying number three of the external requirements (see Table 6-1). The rationale of how this requirement has been met is outlined in Table 6-3

#	Attribute	Rationale for attainment
1	Repeatable	• Tests can be re-run with the same data • VMs are reverted to same state • Use of automation minimises human error in repetition
2	Reproducible	• Framework published in this research • Malware library can be made available (subject to any restrictions) • Source code to be place on Open Research Data archive
3	Testable	• Artefacts are measurable (i.e.: quantifiable observations) • Malware with properties relevant to hypothesis can be selected
4	Controllable	• Test design can select: Tool, Network services, Execution time, Malware by property (e.g.: Network aware, Autostart on boot) • Design allows for use of different VM guest operating system
5	Unbiased	• Malware is randomly selected (with/without a specified property) • Malware can be imported from different sources • Modular design allows for use of different Oracle

Table 6-3 : Rationale for trusted practice attainment

The reproducibility attribute also satisfies the requirement that the theory/technique should be peer reviewed or published (Requirement 5).

Nonetheless, three of the nine external requirements have not been met. Arguably, this renders the implementation of the MATEF only a partial success. However, a counter argument is that the reasons for this are primarily as a result of matters of scope and the fledgling nature of the MATEF as a research project, which we discuss below.

Concerning scope, the validation (Requirement 4, Table 6-1) of any software tool used to examine malware is difficult, but not impossible to achieve. Establishing ground truth concerning the artefacts produced by malware requires multiple forms of analysis concerning the capabilities of malware. Traditionally, this is a labour intensive process and usually reserved only for malware that warrants a deeper understanding of its behaviour, such as seen with Stuxnet (Falliere, Murchu & Chien, 2011). Hence, this level of knowledge about a malware binary (arguably closer to ground truth) is not easily accomplished fully at scale on large numbers of malware, despite attempts to automate the process (Farley, 2015). Therefore, establishing the ground truth concerning malware behaviour is beyond the scope

of the MATEF, which is not a malware analysis tool but instead a framework to evaluate the tools used to perform malware analysis.

The fledgling state of the research impacts on the requirement for admissibility (Requirement 2, Table 6-1), as the MATEF is too new to have been applied to a live investigation that has subsequently gone to court. Consequently, this is not a failing of the framework; it simply remains to be evaluated once the MATEF is applied to evidence submitted to court.

Similarly, it is too early in the project's lifecycle for the MATEF to have been accepted by other practitioners or academics within the community. Hence, the requirement for general acceptance (Requirement 6, Table 6-1), is one that can only be evaluated once there has been an opportunity for the MATEF to be adopted for use by others.

Turning to the internal requirements, summarised in Table 6-2, it can be seen that each of the internal requirements have been met. Almost all of these requirements were satisfied through the design. The framework assumes zero knowledge about the internal operation of the tools tested, thereby satisfying Requirement 1. Other requirements met by design include the extensive use of VMs (Requirement 2), limiting the framework to a single operating system (Requirement 3), providing a malware friendly environment (Requirement 4) and being able to accept malware from any source by simply importing samples from a specified folder (Requirement 5). The need to avoid cross-contamination of malware binaries (Requirement 6) is also met through the design by using encrypted zip files and using scripts to automate the process of deleting all the files in a folder prior to decrypting the binary for use (Requirement 7). The design of the framework also stipulates the folder structure that houses the malware binaries (the *Malware Library,* see section 4.3.2) has restricted permissions allocated to it. Furthermore, the encrypted zip files containing the malware are also password protected (Requirement 8).

The final internal requirement to determine the expected number of artefacts is achieved through both the use of an external system (the *Oracle*, see section 4.3.5) and a design feature whereby an alternative Oracle can be used in the event the chosen one is no longer available.

The above requirements were developed to address the aims of the framework. Hence it is worthwhile also considering how well these aims have been addressed.

Aims

To recap, the aims from section 4.1 were:

1. Use real-world malware
2. Evaluate a tool's ability to detect malware artefacts
3. Mitigate against anti-forensic techniques
4. Produce software product to test tools

The first of the aims was to use real-world malware. This aim is achieved, as the design of the MATEF includes the use of such malware sourced from VirusTotal (2010) (see section 4.3.1).

The second aim was that the MATEF should evaluate a tool's ability to detect malware artefacts. It is argued that the framework also achieves this aim, as it records a given tool's observed number of artefacts against the expected number of artefacts, producing a measure of 'error'. The results from this research indicate that subsequent statistical analysis can also provide a measure of statistical significance when comparing errors from one set of operating conditions to another, e.g.: different execution times.

Consideration for the mitigation against anti-forensic techniques forms the third aim. Whilst completely meeting this aim is outside the scope of this research (see section 2.4), there has been room to mitigate this in part. The discussion on proposed metrics (section 3.3.2) identified the use of quantities of artefacts rather than their values to avoid variation down to random behaviour. Examples include counting the number of files created, instead of recording randomly generated filenames; and counting the quantity of ports opened instead of recording the number (identifier) of the port, which is again highly variable. Hence this aim has been partially met.

The final stated aim of the framework is to produce a software product to implement the framework into a useable product that can be put into practice. This aim has also been met, due to the existence of the code and the research results, available in the Open University Research Data Archive.

Having considered the requirements and aims, it is useful to identify a number of opportunities to develop and improve the MATEF. The first of these is performance.

6.3 Performance evaluation of the MATEF

Two areas where the performance of the MATEF could be evaluated are its speed and resource utilisation.

Speed

The MATEF has been developed from a functionality perspective and so has room to improve the speed with which it produces results. For example, selecting 4,800 malware binaries to run in parallel across 60 VMs translates to each VM being prepared, booted, run and reverted 80 times to execute each malware binary just once. A list of test runs performed is provided in APPENDIX B where it can be seen that executing the malware for 1 minute with Process Monitor, for example, took an average of 15 hours and 26 minutes (or 926 minutes) to complete a Test Run (see Tests 127, 126 and 114). A *Test Run* is defined here as the process of completing the testing of all selected malware binaries, which in this example is 4,800 binaries. This time is reproduced in column 2 of Table 6-4.

To distribute the load on the virtual machine manager, each VM was initially started in a staggered fashion, using an initial delay made up of multiples of 10 seconds (denoted by 'D' in Figure 6-1). This means that VM60 was subject to the longest delay in starting, which was (60-1)*10 = 590 seconds or approximately 10 minutes.

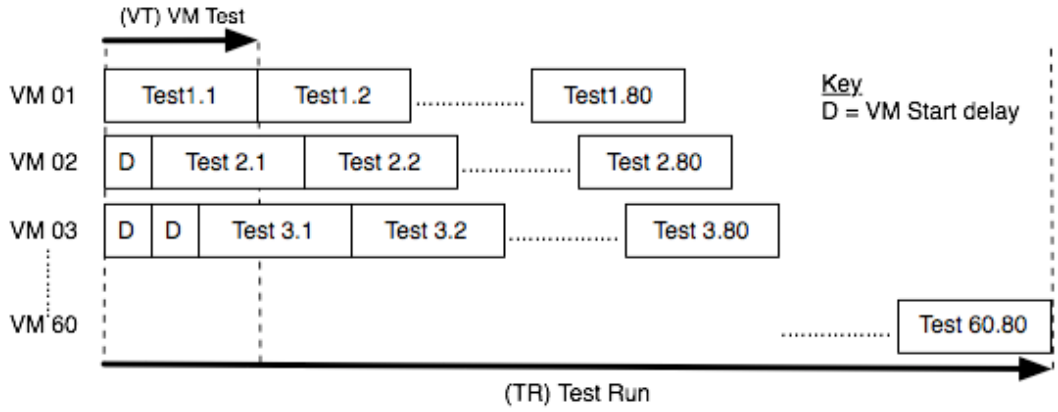

Figure 6-1 : Test Run space for executing 4,800 binaries

The time to complete the Test Run is comprised of this initial maximum delay of 10 minutes plus the time to execute the 80 tests that follow. By subtracting this delay from the total Test Run time, it is possible to calculate the length of each of these individual tests (referred to here as a 'VM Test'), see column 3 of Table 6-4.

To illustrate this, return to the example of using Process Monitor running on VM60 to observe the malware for 1 minute; the Test Run (TR) time is 926 minutes (see above). Hence the time to complete all 80 tests (Test time) is 926 – 10 = 916 minutes, see Figure 6-2.

Given there are 80 VM Tests run sequentially in the Test time, each VM Test therefore requires 916 / 80 = 11.45 minutes to complete, see Row 3, Columns 3 and 4 of Table 6-4.

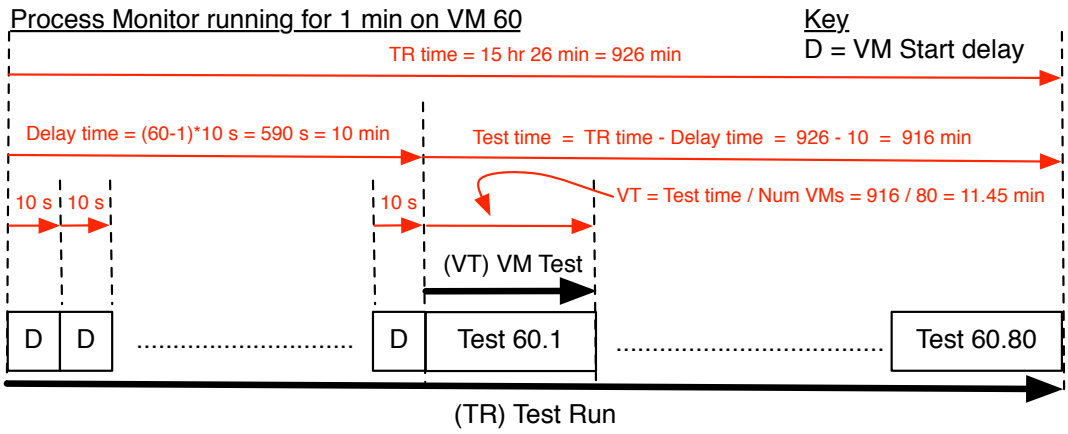

Figure 6-2 : Example timings for Process Monitor on VM60 running for 1 min

Breaking this time down further, each VM Test comprises five high level stages, see Figure 6-3.

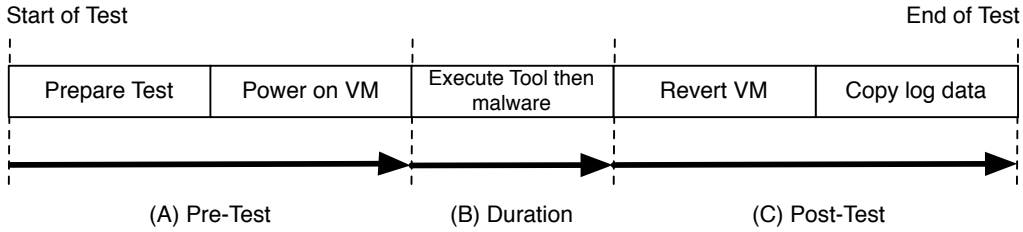

Figure 6-3 : Breakdown of VM Test

The three phases identified in Figure 6-3 are described as follows:

Pre-Test phase (A): This is the time taken to complete the *Prepare Test* and *Power on VM* stages. The former of these identifies the artefacts for the malware under test, locates the folder where the malware is stored in the Malware Library, copies this to the working folder and finally copies the batch files to be used by the VM to control the tool under test (see Pseudocode 2 in section 4.3.4). The latter *Power on VM* stage simply considers the time taken to power on the VM.

Duration phase (B): This is the execution time of the malware, as specified by the user, see column 1 of Table 6-4.

Post-Test phase (C): This is the time taken to complete the *Revert VM* and *copy log data* stages. The former stage (*Power on VM*) simply considers the time taken to revert the VM. The latter stage accounts for the time taken to rename the log file (to a name that identifies the VM and test number) and copy this log file to another folder for subsequent analysis (see Pseudocode 2 in section 4.3.4).

Subtracting the malware execution time (marked 'B' in Figure 6-3) from the VM Test time results in the time required to complete the *Pre-Test* and *Post-Test* phases (i.e.: the sum time for A+C in Figure 6-3). Applying this to the example of Process Monitor being used to observe the malware for 1 minute gives an elapsed time of 11.45 - 1 = 10.45 minutes for this sum time of A+C; see row 3, column 5 of Table 6-4.

		1	2	3	4	5
	Tool	**Execution time of malware**	**Avg Test LENGTH (HH:MM)**	**Avg Test LENGTH (mins)**	**VM Test Time (mins)**	**Pre+Post Time (mins)**
1	Process Monitor	10 sec	13:31	801	10.01	09.85
2	TCPVCon	10 sec	13:11	781	09.76	09.59
3	Process Monitor	01 min	15:26	916	11.45	10.45
4	TCPVCon	01 min	13:09	779	09.74	08.74
5	Process Monitor	05 min	21:44	1294	16.17	11.17
6	TCPVCon	05 min	20:37	1228	15.34	10.34
7	Process Monitor	10 min	22:48	1358	16.98	06.98
8	TCPVCon	10 min	24:17	1447	18.09	08.09

Table 6-4 : Average test times

Note: See the next page for a description of these column headings.

Meaning of column headings:

Column	Description
1	The time duration the malware binary was executed for, depicted by stage B in Figure 6-3.
2	The average of the time taken to complete THREE Test Runs under the same conditions. Represented in HH:MM.
3	As column 2, but accounts for (subtracts) the longest delay in starting VM60, i.e.: 10 minutes. Represented in minutes.
4	The average time taken to complete ONE VM Test, depicted by the sum of the time taken to complete stages A, B and C in Figure 6-3.
5	The time taken to complete stages A and C in Figure 6-3. Calculated by subtracting column 1 from column 4. Represented in minutes.

The *Pre-Test* and *Post-Test* processing time (column 5 of Table 6-4) identifies what is approximately a 10 minute overhead per VM Test, which in Test Runs with large samples, adds a significant cost to the processing time. In the best-case scenario with a zero overhead for a sample size this large, the Test Run time for a 1 minute VM Test would be the initial maximum delay of 10 minutes plus the time to execute the 80 tests (each lasting 1 minute) that follow, ie: 10 + (80 * 1) = 90 minutes.

There are two areas where the VM Test time could be reduced. The first of these would be by reviewing the Prepare Test stage, see Figure 6-3. The code that operates in this stage makes multiple calls to the malware database to retrieve artefact information (see *Pseudocode 2* in section 4.3.4). The code also has a high level of logging in operation that inevitably will have an impact on the speed of code execution. This code could therefore be reviewed and optimized.

The second area where the Test Run time could be reduced would be to increase the number of VMs available for the test. The MATEF is designed to be extensible and could thus be configured to use a larger number of VMs, thereby changing the testing space depicted in Figure 6-1. For the same number of malware binaries to be examined, fewer than 80 VM Tests would need to be conducted if there were more than 60 VMs available.

Alongside these speed issues, recall from above that the MATEF can also be evaluated in terms of its resource utilisation.

Resource utilisation

For the MATEF to operate effectively it requires several resources. In the first instance, it requires a virtualised environment in which to operate. The MATEF uses an open design to make it as portable as possible. Hence although it is currently implemented within a VMWare solution (see section 4.3.6), any virtualised environment that can be automated via scripts will suffice. The MATEF is also extensible, as subject to available resources, additional VMs can be added to improve performance.

Disk space

The disk space requirements for the MATEF are largely subject to the given implementation in place. The installation used for this research consumes disk space as summarised in Table 6.5.

The artefact files comprises a list of all the artefacts expected for all of the malware in a Test Run and the sample path files contain the full path to the malware binary to be extracted. These are generated at the commencement of a Test Run and are formed from multiple queries made to the Malware Database. Originally, this data was queried during the individual VM Tests, but this was found to slow the testing process down, hence these queries are now made in advance of the individual VM Tests.

Element	**Disk space (Total)**	**Disk space (per VM Test)[1]**
Malware binaries[2]	61 GB	-
Malware artefacts DB	310 MB	-
MATEF scripts	< 1 MB	-
Oracle Reports	5.1 GB	-
Tool log files[3]	120 GB	100 MB
Artefact files[4]	-	22 MB
Sample Path Files[5]	-	18 MB
TOTAL	**186.4 GB**	

Table 6-5 : MATEF disk space usage

Notes for Table 6-5:

1 Figures based upon a 10 second VM Test using *TCPVCon*
2 Refers to entire population (over 350,000) held in gzipped files
3 Refers to the gzipped text based log files produced by each tool
4 List of all the artefacts expected for all of the malware in a Test Run
5 Files that contain the full path to the malware binary to be extracted

The Oracle

Another resource the MATEF is dependent on is an Oracle to provide the point of reference for the tools under evaluation. Again, the design of the MATEF is such that it is modular, so if one Oracle becomes unavailable another can be 'plugged in' as an alternative.

Despite this flexibility, the supply of Oracle reports feeding into the MATEF is outside the scope of the MATEF design. The Oracle provider dictates both the rate at which malware binaries can be submitted and the subsequent reports are provided. This is anticipated to be more of a challenge when implementing a fresh installation of the MATEF where the malware used is such that it has not been previously stored on a MATEF framework and hence is not available to be shared. This is less of an issue for an on-going and established implementation with a large population of malware and associated Oracle reports.

However, this issue can be addressed either by developing an in-house Oracle or by integrating an existing solution, such as the Cuckoo Foundation (2016) sandbox. Furthermore, it may be conceivable to establish a Service Level Agreement (SLA) with an existing online malware analysis platform provider.

Maintenance

The MATEF requires several fundamental maintenance operations to perform a test and undertake analysis.

From a testing perspective, there are three fundamental maintenance operations to perform a test and one maintenance operation to undertake analysis. Malware must first be sourced, then submitted to the Oracle and finally imported into the MATEF for subsequent deployment during testing. Depending on any arrangements with the source and the security policy in place at the organisation where the MATEF is implemented, all three of these steps can be automated.

The implementation used for the first of these three operations was such that a feed of malware from the source VirusTotal (2010) was manually enabled for a limited time period and upon request only. Furthermore, delivery of the feed was only possible via email to an externally hosted account, due to security restrictions in place at the University.

Uploading the malware to the Oracle for the second operation was even more challenging for this research, as this essentially meant that the University would effectively be distributing malware outside of its own network, potentially making it liable for any issues that might arise if appropriate measures were not put into place. Hence controls implemented included

the use of encrypted files copied to an Internet facing computer located outside the main network on the University's demilitarised zone (DMZ). Access to this location was restricted via credentials supplied over an SSL connection to a single authorised computer (the Oracle server), which then downloaded the encrypted malware from the university's DMZ server.

The final test related maintenance operation concerns adding a new tool to the MATEF for testing. Alongside the tool binary itself, a DOS batch file must be created to initiate and shutdown the tool (where available) and provide any required command-line arguments. This was found to be a relatively straightforward operation to implement and was only limited by the available command-line options available for the tool.

From an analysis perspective, the MATEF currently requires that the output of the tool must be analysed to identify 'footprints' of given activity. For example, the tool may differentiate and report differently on the creation of a new file that has resulted from a 'Save as' operation compared to a 'File … New' operation. Once understood, the interpretation of the log file must be coded into a Python file, referred to as the tool wrapper (see Section 4.3.8).

In practice, it was not always possible to interpret the output of a tool. With many of the tools having little or no documentation provided and even less (if any) technical support, this could delay (or even prevent) the use of the given tool within the MATEF.

Statistical analysis software

As stated in Section 4.3.8, the statistical analysis component uses a script in conjunction with a wrapper to read the log file of the tool used in the test. The product of this script is a comma separated value (CSV) file.

The current implementation of the MATEF analyses this file using the statistical software SPSS (http://www.ibm.com/spss). This is a manual process and requires skills and knowledge in using SPSS. Furthermore, skills and knowledge in the interpretation of the statistical results are needed to obtain an informed view of the tool that has been tested. Much of this analysis could be automated to remove the dependence on SPSS and possibly some of the manual statistical interpretation of the results.

Having considered the areas where the performance of the MATEF could be improved, an evaluation against the Research Question will now be considered.

6.4 Evaluation against the Research Question

By way of a reminder, the Research Question in section 1.2 stated:

> *Can a systematic basis for trusted practice be established for evaluating malware artefact detection tools used within a forensic investigation?*

How well the MATEF addresses this question can be approached by first recalling from section 3.3 that trusted practice is defined as the trust placed on the reliability of the evidence tendered. Recall further that the Forensic Regulator's Codes of Practice and Conduct (2016) provides a framework by which to establish such practice and is implemented through a number of requirements. As argued in section 6.2, these requirements have largely been met.

In terms of MATEF's ability to evaluate malware artefact detection tools, this has been successful, in that a means to quantify and differentiate the results obtained from different software tools has been achieved. The methodology taken can also be argued to be systematic in nature, as dependent variables have been identified and monitored as a result of making changes to independent variables throughout.

Therefore, the MATEF provides a systematic means by which to evaluate tools and provide data to inform a practitioner's decision in their choice of tools for a forensic investigation involving malware. Given a scientific methodology to obtain this data was previously absent, the MATEF is the first to establish a methodical approach to increase the trust placed in software tools used in the practice of malware forensics.

Despite these positives, there are several opportunities to develop and improve the MATEF. These are explored in the next section covering the limitations of the MATEF.

6.5 Limitations of the MATEF

This section presents what are anticipated as the main criticisms of the research. Where possible, each of the criticisms highlighted is addressed. The latter half of the section presents thoughts on contingencies in the event the primary research direction becomes unattainable.

Representative malware population

The implementation of the MATEF used (see Section 4.5) saw the exclusion of malware that was not repeatable. This might be considered to reduce the representativeness of the malware used to test the tools. Furthermore, it could also be argued that the rapid and relentless growth in malware means the MATEF approach does not consider a malware population that

is representative at the time of testing. However, it is also argued that much of modern malware is adapted from existing code and so radically new behaviour is rare (de la Cuadra, 2007). Alzab (2015) agrees, stating that malware authors are "recycling existing malware" using obfuscation techniques instead of writing new code. A study by Bayer et al. (2009) identified similarities of behaviour between 901,294 samples of malware.

Furthermore, referring to the Aims of the research (Section 1.2), the research seeks to provide a *methodology* to evaluate malware analysis tools, hence updating the malware dataset prior to performing an analysis would address this criticism. Thus the design and utility of the MATEF is independent of the choice of malware used to populate its database.

Reproducibility concerns

It is possible that the results for a given tool will vary between different organisations using this methodology. This is not an uncommon problem and has been identified in a conventional computer forensics context by Garfinkle *et al*. (2009). It is also recognised by the VIM standard (JCGM, 2008), which defines this situation in terms of reproducibility. Thus, rather than being a 'problem', this phenomenon is considered a useful by-product of the framework that would facilitate any future cross-lab study into reproducibility of tools.

Oracle (third-party) dependency

A key element of the MATEF design is the malware database and its representative content. The maintenance of this database is dependent on access to third-party databases storing artefact details in proprietary formats. This reliance on a third-party may be identified as a weakness of the framework. The intention in the design however is to build redundancy into the system by designing the database to be populated from multiple sources, thereby spreading the risk of source availability. The disadvantage of this approach is that some sources provide a richer level of artefact detail than others.

Accuracy of the Oracle

Criticism may also be directed at the accuracy of the third party providing malware artefact information. Online sandbox tools may only execute samples once and for no more than a maximum time duration before terminating (Bayer, Habibi, Balzarotti, Kirda & Kruegel, 2009). Furthermore, malware can typically behave differently each time it is run (Moser, Kruegel & Kirda, 2007) or not run at all if it detects a monitored environment. There is therefore an unknown level of doubt or uncertainty in the accuracy of the artefacts reported by the third-party sandbox tool. This can be addressed to some degree by validating results against well-documented malware samples such as those belonging to the Zeus family.

Testing against zero-day malware

Young or zero-day malware may not be in any of the online source databases at the time a tool is tested. Hence if there is a requirement to test a tool against a specific sample of malware identified on a suspect's computer, this may not be possible until it has been submitted to one or more online analysis engines. Under these circumstances a decision would need to be made to submit it to one or more online sandboxes for analysis. However this decision must be taken in light of the associated risks, such as alerting the malware author of its discovery (see section 4.2.5). If the analysis is being performed around the time of the trial, then this is likely to be many months since the alleged offence. Under these circumstances, the likelihood that the malware is not been reported is much smaller. Alternatively, the sample could be analysed with an offline Oracle such as the Cuckoo Foundation (2016) sandbox.

Having reviewed the different approaches to evaluating the MATEF, these will be brought together to identify further work.

6.6 Evaluation conclusions and further work

In the previous section it has been shown that although the MATEF meets all of the internal requirements (see Table 6-2), three of the external requirements (admissibility, validated and generally accepted) were not met, see Table 6-1. Furthermore, whilst most of the aims were achieved, the anti-forensic mitigations were only partly met. See the discussion under further work below.

As set out below, there is scope for further improvement of the MATEF, however it nonetheless does provide a systematic means by which to evaluate tools and provide data to inform a practitioner's decision in their choice of tools for a forensic investigation involving malware, thereby addressing the Research Question (section 1.2). Before considering the areas for further work it is worth highlighting those areas where it is felt there is little or no room for improvement.

Areas unlikely to be improved

The evaluation has identified some areas of MATEF that are realistically not likely to be improved upon. The inclusion of a command-line interface (CLI) to the large number of existing tools that do not have a CLI for both the execution of the tool and the export of its log file are outside the control of the MATEF.

Another area is the process of adding a new tool, which requires the manual inspection of the tool's output to encode this logic into a wrapper file. The lack of any standard format in the output of similar tools results in a diverse range of output formats from tools, each of which need to be linked to known input events (user actions) to interpret them correctly.

It can be argued that these areas fall more within the implementation of the MATEF rather than the framework itself. Hence their impact on the MATEF is minimal.

Areas that could be improved

In terms of further work, the three areas not met in the requirements are identified in Table 6-1. Briefly, these concern admissibility, validation and general acceptance.

The first of these can be addressed by evaluating tools using the MATEF and then including the results from such tests in the evidence package produced for cases submitted to the Criminal Justice System. If it is determined that the results produced by the MATEF informs the decision made on admissibility, then this requirement can be argued to have been met.

The second requirement, validation, is more challenging to achieve. To validate the output from the MATEF requires a 'ground truth' to compare the results to. Realistically the only sure way to achieve this is to produce one's own software that exhibits the same behaviour as malware in terms of the artefacts it produces and the manner in which they are produced, e.g.: employ the use of anti-forensic techniques in an attempt to hide such artefacts.

The last of these three unmet requirements, general acceptance, is achievable with time if the MATEF is adopted into working practice. As with all new developments, it is difficult to demonstrate wider acceptance until later in the life-cycle of the project.

From a performance perspective, the time to complete a Test Run remains the most significant area for further work to make the MATEF more practical for everyday use. Furthermore, a more economical use of disk space would improve the resource efficiency of the MATEF.

A number of processes currently performed manually could be automated to alleviate the time required to undertake them. These include sourcing and importing malware, performing statistical analysis on the CSV files and interpreting the results. Sourcing might be achieved via the deployment of honeypots or subscriptions to malware share resources, whilst the remainder could be implemented via scripts.

Longer term, the MATEF could be developed to cater for graphical user interface (GUI) tools. Arguably this is more of a limitation of a tool rather than the MATEF if the tool cannot

be automated via a script. Another long term development may be the introduction of more recent operating systems into the VMs as a test environment. Although more of an implementation (rather than framework) development, this would present challenge on two fronts. The first being that when trying to test a tool, as many of the tools will not operate in more recent operating systems. Secondly, the current operating system used by the MATEF implementation (Windows XP) has been selected given it meets the external requirement to be a fertile environment for testing malware, see section 3.3.1.

6.7 Chapter summary

This chapter evaluated the MATEF from a number of different perspectives. To facilitate this, it opened by first considering the evaluation criteria that can be applied in section 6.1. This identified that one approach to this is to evaluate how well MATEF has met the requirements of the framework, as well as how well it has achieved the aims of the framework, see section 6.2. In addition to addressing the requirements and aims, the performance measures of the MATEF were identified and discussed in section 6.3. This section largely considered the speed and resource requirements of the MATEF. Evaluating the MATEF against the requirements, aims and performance measures provides grounding for establishing how well the MATEF has addressed the fundamental motivation for the research; expressed through the Research Question, see section 6.4. The chapter drew to a close by first identifying the limitations of the MATEF and responding to each of these in turn. Following this, the chapter synthesised the findings of the above critique and presented a discussion on further work.

The next chapter recaps on the preceding chapters and draws conclusions on the thesis as a whole.

Chapter 7 Conclusions

We live in an increasingly interconnected world where technology is ubiquitous and much of our infrastructure and economy is dependent on our ability to operate in cyber space. Since 2011 the UK Government has continued to invest in developing the UK's resilience to the cyber threat, labelled as a "Tier One risk to UK interests" (Cabinet Office, 2016). This requirement to develop the UK's capability applies not just to national security but, as Burd *et. al*. (2011) argue, to cybercrime as well.

This research has reported how changes to how cybercrime investigations are conducted within the UK Criminal Justice System (CJS) have identified a number of factors that have led to challenges to some expert evidence submitted to criminal proceedings. Factors such as the 'Trojan defence', unfounded trust in software tools, problems with expert evidence and lack of provenance are all areas where evidence submitted is open to challenge. Furthermore, the now active requirement for forensic practitioners (including teams operating within the police) to be accredited by the Forensic Science Regulator in order to submit evidence to the CJS, means practitioners need to evidence their trust in tools used for investigations, including those involving malware.

To address this requirement, a framework has been developed to provide empirical data on the ability of software tools to identify artefacts produced by malware. To summarise the success of his framework, it is worth recapping on the research goals, which this framework addresses.

7.1 Goals and findings

By way of recap, the goals of the research were identified in Table 1-2 (Section 1.2) as:

- Determine if there is there a problem with a lack of trusted practice in malware forensics
- Identify the requirements for a solution
- Develop a methodology for evaluating malware artefact detection tools

It is argued that the first of these goals has been met, as evidenced by the literature review (Chapter 2). This has provided a case for a lack of trusted practice in the field of malware forensics.

The second goal has also been met; given the research identified a number of requirements to develop the framework (see Chapter 3) and the majority of these were met (see Chapter 6).

The third goal is perhaps the most significant of the goals. It is argued that this goal of the research has also been met, as is demonstrated by the empirical evidence produced using the MATEF. A notable aspect of this research is that the framework has been implemented and tested using a large population of real-world malware binaries (over of 350,000). This is relatively large when compared to other research groups who use fewer numbers of malware binaries. For example, Gashi *et al*. (2009) used 1,599 malware binaries during their study on anti-malware engines, whilst in a study (Gashi, Sobesto, Stankovic & Cukier, 2013) they used less than half this amount (900). Zolkipli and Jantan (2011) used a sample size of just 5 binaries in their study on malware behaviour.

Despite the success of this research, it is recognised that there are areas for improvement.

7.2 Critical review of thesis

Two primary criticisms can be levelled at this thesis. These are a limitation in terms of (a) scope and (b) methodology. These areas will be discussed in the following sub-sections.

7.2.1 Scope limitations

This thesis has been limited by its scope to:

1. Quantity of artefacts observed when evaluating tools and not their values.
2. Support for only evaluating Command-line interface (CLI) tools

This limited scope of the thesis may suggest that its findings are limited as well. However, the reduced information available from observing quantities and not values does not prevent a comparison being made between the expected and observed quantities. Furthermore, this approach reduces the number of random variables from 2 to 1, as any variation in values is ignored by this approach.

The support for CLI tools only again does not prevent a comparison being made between the expected and observed quantities, as at this time it only restricts which tools can be evaluated in this way. In a time of growing quantities of data to process, the lack of command-line support by a tool to facilitate automation is more of a limitation of the tool than of the MATEF.

Despite these limitations in scope, the ability to compare expected and observed values is still honoured and hence satisfies the Research Question. Furthermore, this research is still useful in that the findings:

- Provide a means by which to evaluate tools in a malware environment, where previously there was none.
- Add to common body of knowledge on software evaluation within the relatively young malware forensics field.

7.2.2 Methodology limitations

Critics of the MATEF may argue the use of another tool (in this case an online sandbox) to determine the expected numbers of artefacts does not provide an accurate representation of the true numbers of artefacts to be expected from executing a given malware binary. Others may go further and suggest that it is not possible to obtain such a figure, due to the random nature of the malware.

In response to these criticisms, random variations that manifest themselves when determining an approximated ground truth (as discussed in section 4.2.5) are inherently challenging to overcome. However, the effect of these variations are minimised by performing multiple test runs and taking an average of the number of observed artefacts, see Table 6-4. Furthermore, Hubbard (2014, p. 162) points out that if there is a lot of uncertainty in a quantity, then very little data is needed to reduce the uncertainty significantly. In other words, gaining a little knowledge about how a tool copes with observing malware where previously there was a high level of uncertainty is a significant advance in our understanding of that tool. Hence, producing an estimate of the expected number of artefacts to be observed significantly reduces the uncertainty in what is expected from subsequent observations. Consequently, a reduction in uncertainty leads to an increase in trust (Bell, 2017, p. xix), hence this approach addresses the Research Question.

7.3 Contributions

This research contributes to the common body of knowledge in the area of software tool evaluation in a malware forensics setting. The main contribution is that it is the first to provide a framework to facilitate the empirical evaluation of a tool's ability to detect malware artefacts under different operating conditions. To recap from section 1.2, the Research Question states:

> *Can a systematic basis for trusted practice be established for evaluating malware artefact detection tools used within a forensic investigation?*

To address this question the following related sub-questions were investigated:

1. To what extent is there a case for a lack of trusted practice?
2. What are the requirements for evaluating malware artefact detection tools?
3. Do the conditions under which tools and malware operate have an effect on the ability to observe malware behaviour?
4. Are observations of malware behaviour impacted by the practitioner's choice of tool?
5. What factors can be used to evaluate the performance of the methodology and hence identify areas of improvement.

To begin with, exploratory evidence in the literature review (Chapter 2) has provided a case for a lack of trusted practice in the field of malware forensics (research sub-question 1 above).

Another contribution is the systematic identification of a set of requirements for establishing trusted practice in the use of malware artefact detection tools (research sub-question 2 above).

Two further contributions come from empirical evidence generated by the tools tested during this research. The first of these compares how two different tools operate under different conditions (research sub-question 3 above), identifying an optimal execution time for a given tool. Secondly, empirical data is provided showing how these tools perform when compared with each other under the same operating conditions (research sub-question 4 above).

An additional contribution is provided from the empirical evidence gathered on the performance of this framework, enabling areas of improvement to be identified (research sub-question 5 above).

More generally, the MATEF provides a systematic methodology for practitioners to apply to new or unfamiliar tools that will allow them to specify parameters, such as how long the tool must be run for to obtain the optimal number of artefacts.

In summary, the contributions of this thesis can be summed up to be:

1. Confirmation for case for a lack of trusted practice in the field of malware forensics, evidence from the literature review.
2. A framework to facilitate the production of empirical evidence of a tool's ability to detect malware artefacts under different operating conditions, evidenced by the design and implementation of the MATEF, Chapter 4
3. A set of requirements for establishing trusted practice in the use of malware artefact detection tools, evidenced by Chapter 3
4. Empirical evidence generated identifying the optimal execution time for a given tool when observing malware artefacts, evidenced by Chapter 5
5. Empirical evidence that the choice of tool can impact on the number of artefacts observed, evidenced by Chapter 5
6. Empirical evidence of the performance of this framework, evidenced by section 6.3.
7. A systematic methodology for practitioners to specify operating parameters (such as how long the tool must be run for) when obtaining new or unfamiliar tools.

7.4 Further work

The evaluation of the research (see section 6.6) identified a number of areas for further work. The first of these was that if the MATEF was extended to include its own Oracle analysis platform the issue surrounding the rate of submission of malware to and subsequent delivery of reports would be overcome. Another area identified that would extend the scope of the MATEF significantly would be the support for tools that do not support a CLI. Also proposed for further work were the admissibility, validation and general acceptance requirements (see Table 6-1) that were not met by this research.

Performance issues were also identified as areas where further work could be undertaken. The time to complete a Test Run and a more economical use of disk space were singled out as specific areas of improvement.

The use of both bare metal and virtual machines together to test malware analysis tools is a recommended malware analysis lab requirement (see section 3.3.2) and would be of benefit for testing tools where the malware binary is aware of a virtualised environment and so behaves differently. However, consideration should be given to the impact this approach would have on the speed of testing, which would be slower to allow physical machines to be reset between tests.

The process of evaluating a malware analysis tool to observe malware on different operating systems is not an extension of the MATEF; this is because the framework is conceptually the same, regardless of the implementation. However, by implementing the framework on different VM platforms in parallel, the results obtained will more inclusive and further help to inform the practitioner in their choice of tool, regardless of the operating system in place on a suspect's computer.

A number of processes that are currently manually performed were also considered for further work. These include sourcing and ingesting malware into the MATEF platform, statistical analysis operations on CSV files derived from tool log files and interpreting the results.

7.5 Chapter summary

This chapter opened by revisiting the goals of the research and considering the extent to which these have been met. Of particular note was the size of the dataset used in the research, which is significantly larger than those used in other studies.

A critique of the thesis followed and examined the scope and methodologies of the research. A case for using quantities rather than values in observations was made to minimise the effects of random variations and thereby increase the level of trust in the data. The issue of supporting only command-line interface (CLI) tools was also discussed. Whilst acknowledging the limits this placed on the MATEF's scope it was argued this was more of a limitation of the tools being tested than the MATEF itself.

The difficulties of determining 'Ground Truth' were identified and the method used to estimate this discussed, concluding that the approach reduces uncertainty and thereby increase trust in the results obtained.

The chapter closed with a review and summary of the contributions made by this research, followed by suggestions for further work.

7.6 Concluding remarks

In this thesis we have provided a case for a lack of trusted practice in the field of malware forensics. To address this, we identified the gap between current practice and the regulatory, legal and technical requirements. We further went on to design a framework designed to systematically address the gap and apply scientific principles to the testing of malware analysis tools. A prototype was built to implement the framework and used to test tools on a large corpus of malware.

Whilst it is acknowledged there are limitations of the prototype implemented in terms of scope and the establishment of ground truth (see section 7.2), such limitations do not affect the framework itself. Indeed, since there is no generally accepted scientific methodology to evaluate tools used in malware analysis, we believe that the work presented in this thesis to develop a framework based on such methodology goes some way towards addressing the lack of trust in tools used in the field of malware forensics. Moreover, the empirical data presented in the thesis has highlighted the optimal execution time of a tool under test. Hence, controls such as this can inform a subsequent procedure, which is then arguably underpinned with scientifically established empirical data.

Furthermore, providing a methodology to evaluate a malware analysis tool where previously there was none goes some way to reducing the uncertainty in the output of the tool. A reduction in uncertainty, in turn, increases the trust placed in the practice of using that tool.

References

Alazab, Mamoun (2015) 'Profiling and classifying the behavior of malicious codes', *Journal of Systems and Software*, 100, pp. 91–102.

Aldeid.com (2017) 'PEiD - aldeid', [online] Available from: https://www.aldeid.com/wiki/PEiD (Accessed 15 February 2017).

Angrishi, Kishore (2017) 'Turning Internet of Things(IoT) into Internet of Vulnerabilities (IoV) : IoT Botnets', *arXiv:1702.03681 [cs]*, [online] Available from: http://arxiv.org/abs/1702.03681 (Accessed 4 March 2017).

Anubis (2010) 'Anubis: Analyzing Unknown Binaries', *Anubis: Analyzing Unjnown Binaries*, [online] Available from: http://anubis.iseclab.org/ (Accessed 13 July 2010).

Appel, E. and Pollitt, M. (2005) *Report on the Digital Evidence Needs Survey of State, Local and Tribal Law Enforcement*, Washington, D.C., National Institute of Justice, [online] Available from: www.jciac.org/docs/Digital%20Evidence%20Survey%20Report.pdf.

Asbury, Michael (2015) 'NASA's IV&V Facility', *NASA*, Text, [online] Available from: http://www.nasa.gov/centers/ivv/home/index.html (Accessed 4 April 2016).

Ashford, Warwick (2010) 'Malware growth reaches record rate', [online] Available from: http://www.computerweekly.com/news/1280094367/Malware-growth-reaches-record-rate (Accessed 7 April 2016).

Aycock, J. (2006) *Computer Viruses and Malware*, Advances in Information Security, Springer.

Ayers, Daniel (2009) 'A second generation computer forensic analysis system', *Digital Investigation*, The Proceedings of the Ninth Annual DFRWS Conference, 6, Supplement, pp. S34–S42.

Baker, Stewart, Filipiak, Natalia and Timlin, Katrina (2011) *In the Dark: Crucial Industries Confront Cyberattacks*, Critical Infrastructure Report, McAfee, [online] Available from: https://www.mcafee.com/us/resources/reports/rp-critical-infrastructure-protection.pdf (Accessed 21 February 2017).

Balaji, S. and Murugaiyan, M. Sundararajan (2012) 'Waterfall vs. V-Model vs. Agile: A comparative study on SDLC', *International Journal of Information Technology and Business Management*, 2(1), pp. 26–30.

Bayer, Ulrich, Habibi, Imam, Balzarotti, Davide, Kirda, Engin and Kruegel, Christopher (2009) 'A view on current malware behaviors', In *Proceedings of the 2nd USENIX conference on Large-scale exploits and emergent threats: botnets, spyware, worms, and more*, LEET'09, Berkeley, CA, USA, USENIX Association, pp. 8–8, [online] Available from: http://portal.acm.org/citation.cfm?id=1855676.1855684.

Bayer, Ulrich, Moser, Andreas, Kruegel, Christopher and Kirda, Engin (2006) 'Dynamic Analysis of Malicious Code', *Journal in Computer Virology*, 2(1), pp. 67–77.

Beach, Jonathan (2010) 'Scientific evidence: a need for caution in decision-making', *Australian Journal of Forensic Sciences*, 42(1), pp. 49–77.

Beckett, Jason (2010) 'Forensic Computing: A Deterministic Model for Validation and Verification through an Ontological Examination of Forensic Functions and Processes', PhD, University of South Australia, [online] Available from: Personal communication from author, September 2011.

Beckett, Jason and Slay, Jill (2007) 'Digital Forensics: Validation and Verification in a Dynamic Work Environment', In *System Sciences, 2007. HICSS 2007. 40th Annual Hawaii International Conference on*, p. 266a–266a.

Bell, Suzanne (2017) *Measurement Uncertainty in Forensic Science: A Practical Guide*, Boca Raton, CRC Press.

Binu, A. and Kumar, G. Santhosh (2011) 'Virtualization Techniques: A Methodical Review of XEN and KVM', In *Advances in Computing and Communications*, Springer, Berlin, Heidelberg, pp. 399–410, [online] Available from: https://link.springer.com/chapter/10.1007/978-3-642-22709-7_40 (Accessed 14 March 2017).

Bloom, James D. (2017) 'Mock Server', *Mock Server*, [online] Available from: http://www.mock-server.com (Accessed 4 March 2017).

Boehm, Barry (1989) 'Software risk management', In Ghezzi, C. and McDermid, J. A. (eds.), *ESEC '89*, Lecture Notes in Computer Science, Springer Berlin Heidelberg, pp. 1–19, [online] Available from: http://link.springer.com/chapter/10.1007/3-540-51635-2_29 (Accessed 31 March 2016).

Boley, Lance (2014) 'Emulation and virtualization: What's the difference?', *DELL - Power More*, Dell blogsite, [online] Available from: https://powermore.dell.com/technology/emulation-virtualization-whats-difference/ (Accessed 22 March 2016).

Bowles, Stephen and Hernandez-Castro, Julio (2015) 'The first 10 years of the Trojan Horse defence', *Computer Fraud & Security*, 2015(1), pp. 5–13.

Bridges, Lloyd (2008) 'The changing face of malware', *Network Security*, 2008(1), pp. 17–20.

British Computer Society (2011) *Joint response from the IET, The Royal Academy of Engineering and BCS, the Chartered Institute for IT to the House of Commons Science and Technology Committee: Inquiry into Malware and Cybercrime*, British Computer Society, [online] Available from: http://www.bcs.org/content/conWebDoc/41596 (Accessed 10 January 2011).

Brown, Cameron SD (2015) 'Investigating and Prosecuting Cyber Crime: Forensic Dependencies and Barriers to Justice', *International Journal of Cyber Criminology*, 9(1), p. 55.

Brown, Christopher (2010) *Computer Evidence*, Cengage Learning.

Burd, S.D., Jones, D.E. and Seazzu, A.F. (2011) 'Bridging Differences in Digital Forensics for Law Enforcement and National Security', In *2011 44th Hawaii International Conference on System Sciences (HICSS)*, pp. 1–6.

Bureau, Pierre-Marc and Harley, David (2008) 'A dose by any other name', In *18th Virus Bulletin International Conference*, Ottawa, Canada, Virus Bulletin Ltd., pp. 224–231.

Byers, David and Shahmehri, Nahid (2009) 'A systematic evaluation of disk imaging in EnCase® 6.8 and LinEn 6.1', *Digital Investigation*, 6(1–2), pp. 61–70.

Cabinet Office (2011) 'Cybersecurity Strategy', *Cabinet Office*, [online] Available from: http://www.cabinetoffice.gov.uk/resource-library/cyber-security-strategy (Accessed 7 December 2011).

Cabinet Office (2016) *National Cybersecurity Strategy 2016 to 2021 - Publications - GOV.UK*, Policy Paper, London, HM Treasury, [online] Available from: https://www.gov.uk/government/publications/national-cyber-security-strategy-2016-to-2021 (Accessed 18 January 2017).

Cabinet Office (2010) 'The national security strategy - a strong Britain in an age of uncertainty - Publications - GOV.UK', [online] Available from: https://www.gov.uk/government/publications/the-national-security-strategy-a-strong-britain-in-an-age-of-uncertainty (Accessed 9 November 2015).

Carrier, B. (2002) 'Open source digital forensics tools: The legal argument', *Stake Research Report*.

Carrier, Brian (2003) 'Defining digital forensic examination and analysis tools using abstraction layers', *International Journal of Digital Evidence*, 1(4).

Carrier, Brian (2010) 'Digital (Computer) Forensics Tool Testing Images', *Digital Forensics Tool Testing Images*, [online] Available from: http://dftt.sourceforge.net/ (Accessed 23 March 2016).

Carrier, Brian D. (2006) 'A hypothesis-based approach to digital forensic investigations', Ph.D., United States -- Indiana, Purdue University, [online] Available from: http://search.proquest.com/docview/305266774 (Accessed 28 April 2016).

Carvey, Harlan (2012) *Windows Forensic Analysis Toolkit, Third Edition: Advanced Analysis Techniques for Windows 7*, 3 edition. Waltham, MA, Syngress.

Carvey, Harlan (2009) 'Windows Incident Response: The Trojan Defense', [online] Available from: http://windowsir.blogspot.com/2009/12/trojan-defense.html (Accessed 22 January 2011).

Casey, E. (2002) 'Error, uncertainty, and loss in digital evidence', *International Journal of Digital Evidence*, 1(2), pp. 1–45.

Casey, Eoghan (2011a) *Digital Evidence and Computer Crime: Forensic Science, Computers, and the Internet*, 3 edition. Waltham, MA, Academic Press.

Casey, Eoghan (2012) 'Editorial - Cutting the Gordian knot: Defining requirements for trustworthy tools', *Digital Investigation*, 8(3–4), pp. 145–146.

Casey, Eoghan (2011b) 'The increasing need for automation and validation in digital forensics', *Digital Investigation*, 7(3–4), pp. 103–104.

Chen, Xu, Andersen, J., Mao, Z. M., Bailey, M. and Nazario, J. (2008) 'Towards an understanding of anti-virtualization and anti-debugging behavior in modern malware', In *IEEE International Conference on Dependable Systems and Networks With FTCS and DCC, 2008. DSN 2008*, pp. 177–186.

Christensen, Angi M., Crowder, Christian M., Ousley, Stephen D. and Houck, Max M. (2014) 'Error and its Meaning in Forensic Science', *Journal of Forensic Sciences*, 59(1), pp. 123–126.

Cole, Simon A. (2011) 'Forensic Science and Wrongful Convictions: From Exposer to Contributor to Corrector', *New England Law Review*, 46, p. 711.

Collier, Ken W. (2011) *Agile Analytics: A Value-Driven Approach to Business Intelligence and Data Warehousing: Delivering the Promise of Business Intelligence*, 1 edition. Upper Saddle River, NJ, Addison Wesley.

Common Criteria (2016) 'Common Criteria : New CC Portal', [online] Available from: https://www.commoncriteriaportal.org/ (Accessed 9 April 2016).

Comodo Group (n.d.) 'Advanced File Analysis System | Valkyrie', [online] Available from: https://valkyrie.comodo.com/ (Accessed 23 August 2016).

Coogan, K., Debray, S., Kaochar, T. and Townsend, G. (2009) 'Automatic Static Unpacking of Malware Binaries', In *16th Working Conference on Reverse Engineering, 2009. WCRE '09*, pp. 167–176.

Cooley, C. M. (2004) 'Reforming the Forensic Science Community to Avert the Ultimate Injustice', *Stanford Law & Policy Review*, 15, p. 381.

Cornish, Paul, Livingstone, David, Clemente, Dave and Yorke, Claire (2011) *Cybersecurity and the UK's Critical National Infrastructure*, Chatham House, [online] Available from: http://www.chathamhouse.org/publications/papers/view/178171 (Accessed 26 November 2011).

Corregedor, M. and Von Solms, S. (2012) 'ATE: Anti-malware technique evaluator', In *Proceedings of the ISSA 2012 Conference*, Johannesburg, South Africa, IEEE, pp. 1–8.

Cottrell, Stella (2014) *Dissertations and Project Reports: A Step by Step Guide*, 2014 ed. edition. Palgrave Macmillan.

CPS (2014) 'Evidence from Computer Records: Legal Guidance: The Crown Prosecution Service', *The Crown Prosecution Service*, [online] Available from: http://www.cps.gov.uk/legal/a_to_c/computer_records_evidence/ (Accessed 23 May 2016).

CPS (2015) 'Guidance on Expert Evidence', Crown Prosecution Service, [online] Available from: http://www.cps.gov.uk/legal/assets/uploads/files/expert_evidence_first_edition_2014.pdf.

de la Cuadra, Fernando (2007) 'The geneology of malware', *Network Security*, 2007(4), pp. 17–20.

Cuckoo Foundation (2016) 'Automated Malware Analysis - Cuckoo Sandbox', [online] Available from: https://cuckoosandbox.org/ (Accessed 14 March 2016).

Cusack, Brian and Liang, James (2011) 'Comparing the Performance of Three Digital Forensic Tools', *Journal of Applied Computing and Information Technology*, 15(1), p. A11.

Cy4or (2009) 'Computer Forensic Analysis & Computer Fraud Investigations Experts', *Cy4or*, [online] Available from: http://www.cy4or.co.uk/forensic-services/computer-forensics (Accessed 8 December 2011).

Daly, T. and Burns, L. (2010) 'Concurrent Architecture for Automated Malware Classification', In *2010 43rd Hawaii International Conference on System Sciences (HICSS)*, pp. 1–8.

DC3 (2016) 'DC3 | Tool Validations', *DoD Defence Cyber Crime Center (DC3)*, [online] Available from: http://www.dc3.mil/tool-validations (Accessed 1 April 2016).

Denning, Peter J (2005) 'Is computer science science?', *Communications of the ACM*, 48(4), pp. 27–31.

Digital Corpora (2017) 'Digital Corpora', *Digital Corpora*, [online] Available from: http://digitalcorpora.org/ (Accessed 15 February 2017).

Douglas, John (2007) 'Trojan defence: the old chestnut...', *Digital Detective*, Closed Law Enforcement forum, [online] Available from: http://www.digital-detective.co.uk/cgi-bin/digitalboard/YaBB.pl?num=1191330237/15 (Accessed 8 April 2011).

Dow, Dennis. M (2007) 'Calibration of Computer Forensic Equipment', MSc, University College University of Denver.

Drinkwater, Richard (2009) 'Forensics from the sausage factory: Facebook revisited and other chat related stuff', *Forensics from the sausage factory*, [online] Available from: http://forensicsfromthesausagefactory.blogspot.com/2009/04/facebook-revisited-and-other-chat.html (Accessed 8 December 2011).

Duranti, Luciana and Rogers, Corinne (2012) 'Trust in digital records: An increasingly cloudy legal area', *Computer Law & Security Review*, 28(5), pp. 522–531.

Dykstra, Josiah and Sherman, Alan T. (2012) 'Acquiring forensic evidence from infrastructure-as-a-service cloud computing: Exploring and evaluating tools, trust, and techniques', *Digital Investigation*, The Proceedings of the Twelfth Annual DFRWS Conference12th Annual Digital Forensics Research Conference, 9, Supplement, pp. S90–S98.

Edmond, G., Biber, K., Kemp, R. I. and Porter, G. (2009) 'Law's Looking Glass: Expert Identification Evidence Derived from Photographic and Video Images', *Current Issues in Criminal Justice*, 20(3).

Egele, M., Scholte, T., Kirda, E. and Kruegel, C. (2012) 'A survey on automated dynamic malware analysis techniques and tools', *ACM Computing Surveys*, 44(2).

Elisan, Christopher C. (2015) *Advanced Malware Analysis*, McGraw-Hill Osborne.

Epstein, Robert (2009) 'Attacking the Government's "Junk Science"', Seminar, Charlston, South Carolina, USA, [online] Available from: http://www.fd.org/pdf_lib/WinningStrategies2009/Attacking_the_Gov_Junk_Science.pdf (Accessed 11 December 2011).

European Union (2009) 'EUR-Lex - 32009F0905 - EN - EUR-Lex', *Eur-Lex*, European law website, [online] Available from: http://eur-lex.europa.eu/legal-

content/EN/NOT/?uri=CELEX:32009F0905&qid=1406753326057 (Accessed 30 July 2014).

Fab4 (2011) 'Open Source (UK Judge) - Digital Forensics Forums | ForensicFocus.com', *Forensic Focus*, [online] Available from: http://www.forensicfocus.com/Forums/viewtopic/p=6550261/ (Accessed 9 April 2016).

Falliere, Nicolas, Murchu, Liam and Chien, Eric (2011) *W32.Stuxnet Dossier*, Security Response, White paper, Symantec, [online] Available from: http://www.symantec.com/content/en/us/enterprise/media/security_response/whitepapers/w32_stuxnet_dossier.pdf.

Farley, Ryan J. (2015) 'Toward Automated Forensic Analysis of Obfuscated Malware', Ph.D., United States -- Virginia, George Mason University, [online] Available from: http://search.proquest.com.libezproxy.open.ac.uk/pqdt/docview/1705878547/abstract/64AC3EA81EBF4661PQ/1 (Accessed 21 March 2016).

FDA (2002) 'Guidance Documents (Medical Devices and Radiation-Emitting Products) - General Principles of Software Validation; Final Guidance for Industry and FDA Staff', U.S. Food and Drug Administration, [online] Available from: http://www.fda.gov/MedicalDevices/DeviceRegulationandGuidance/GuidanceDocuments/ucm085281.htm (Accessed 4 April 2016).

Field, Andy (2013) *Discovering Statistics Using IBM SPSS Statistics*, 4th Revised edition edition. SAGE Publications Ltd.

FireEye (2017) 'ApateDNS', *FireEye*, [online] Available from: https://www.fireeye.com/services/freeware/apatedns.html (Accessed 4 March 2017).

Flandrin, Flavien, Buchanan, William J., Macfarlane, Richard, Ramsay, Bruce and Smales, Adrian (2014) 'Evaluating Digital Forensic Tools (DFTs).', In *7th International Conference : Cybercrime Forensics Education & Training*, Canterbury Christ Church University, Canterbury, England, [online] Available from: http://researchrepository.napier.ac.uk/6906/ (Accessed 12 December 2015).

Ford, R. and Carvalho, M. (2014) 'A significant improvement for anti-malware tests', In *2014 2nd Workshop on Anti-Malware Testing Research (WATeR)*, Canterbury. UK, IEEE, pp. 1–4.

Forensic control (2011) 'What is IT forensics?', *Forensic Control*, [online] Available from: http://forensiccontrol.com/resources/beginners-guide-computer-forensics/ (Accessed 8 December 2011).

Forensic Science Regulator (2015) 'Digital forensics method validation: draft guidance (second consultation) - Consultations - GOV.UK', Home Office, [online] Available from: https://www.gov.uk/government/consultations/digital-forensics-method-validation-draft-guidance-second-consultation (Accessed 11 April 2016).

Forensic Science Regulator (2011) 'Forensic science providers: codes of practice and conduct - Publications - GOV.UK', [online] Available from: https://www.gov.uk/government/publications/forensic-science-providers-codes-of-practice-and-conduct (Accessed 29 July 2014).

Forensic Science Regulator (2016) ‘Forensic science providers: codes of practice and conduct, issue 3’, Crown, [online] Available from: https://www.gov.uk/government/publications/forensic-science-providers-codes-of-practice-and-conduct-2016.

F-Secure (2011) ‘F-Secure Sample Analysis System’, *F-Secure Sample Analysis System*, [online] Available from: https://analysis.f-secure.com/portal/login.html (Accessed 8 April 2011).

G Data Software AG (2016) *G DATA PC Malware Report*, Bochum, Germany, G Data Software AG, [online] Available from: https://file.gdatasoftware.com/web/en/documents/whitepaper/G_DATA_PC_Malware_Report_Jul-Dec_2015_English.pdf.

Gallop, Angela and Brown, Jennifer (2014) ‘The Market Future for Forensic Science Services in England and Wales’, *Policing*, 8(3), pp. 254–264.

Garfinkel, Simson, Farrell, Paul, Roussev, Vassil and Dinolt, George (2009) ‘Bringing science to digital forensics with standardized forensic corpora’, *Digital Investigation*, 6, pp. S2–S11.

Gashi, I., Stankovic, V., Leita, C. and Thonnard, O. (2009) ‘An experimental study of diversity with off-the-shelf antivirus engines’, In Gashi, I., Stankovic, V., Leita, C. and Thonnard, O. (2009) ‘An experimental study of diversity with off-the-shelf antivirus engines’, In *Proceedings of the 8th IEEE International Symposium on Network Computing and Applications,* NCA 2009, Cambridge, MA USA, pp. 4–11.

Gashi, Ilir, Sobesto, Bertrand, Stankovic, Vladimir and Cukier, Michel (2013) ‘Does Malware Detection Improve with Diverse AntiVirus Products? An Empirical Study’, In Bitsch, F., Guiochet, J., and Kaâniche, M. (eds.), *Computer Safety, Reliability, and Security*, Lecture Notes in Computer Science, Springer Berlin Heidelberg, pp. 94–105, [online] Available from: http://link.springer.com.libezproxy.open.ac.uk/chapter/10.1007/978-3-642-40793-2_9 (Accessed 9 April 2015).

GetReading (2003) ‘Program put child porn pics on my PC’, News, [online] Available from: http://www.getreading.co.uk/news/local-news/program-put-child-porn-pics-4271386 (Accessed 9 November 2015).

Gibb, Frances (2003) ‘Virus planted porn on innocent man’s screen’, *The Times*, London, England, 18th April.

Google (2016a) ‘Google Scholar search for “A road map for digital forensic research”’, *Google Scholar*, [online] Available from: https://scholar.google.co.uk/scholar?cites=18912459844865481 73&as_sdt=2005&sciodt=0,5&hl=en (Accessed 8 March 2016).

Google (2016b) ‘Google Scholar search for “Defining Digital Forensic Examination and Analysis Tools Using Abstraction Layers”’, *Google Scholar*, [online] Available from: https://scholar.google.co.uk/scholar?q=%22Defining+Digital+Forensic+Examination+and+Analysis+Tools+Using+Abstraction+Layers%22&hl=en&as_sdt=0%2C5&as_ylo=2003&as_yhi=2003 (Accessed 8 March 2016).

Grant, Ian (2010) ‘Untitled’, *Computer Weekly*, p. 54.

Grégio, André Ricardo Abed, Afonso, Vitor Monte, Filho, Dario Simões Fernandes, Geus, Paulo Lício de and Jino, Mario (2015) 'Toward a Taxonomy of Malware Behaviors', *The Computer Journal*, 58(10), pp. 2758–2777.

Guidance Software Inc. (2011) 'EnCase Legal Journal 2011', Guidance Software Inc., [online] Available from: http://www.guidancesoftware.com/Document.aspx?did=1000017380&id=2525 (Accessed 22 December 2011).

Guo, Y., Slay, J. and Beckett, J. (2009) 'Validation and verification of computer forensic software tools-Searching Function', *Digital Investigation*, 6(SUPPL.), pp. S12–S22.

Guo, Yinghua and Slay, Jill (2010a) 'A Function Oriented Methodology to Validate and Verify Forensic Copy Function of Digital Forensic Tools', In *Availability, Reliability, and Security, 2010. ARES '10 International Conference on*, pp. 665–670.

Guo, Yinghua and Slay, Jill (2010b) 'Computer Forensic Function Testing: Media Preparation, Write Protection And Verification', *Journal of Digital Forensics, Security and Law*, 5(2), pp. 5–20.

Guo, Yinghua and Slay, Jill (2010c) 'Data Recovery Function Testing for Digital Forensic Tools', In Chow, K.-P. and Shenoi, S. (eds.), *Advances in Digital Forensics VI*, IFIP Advances in Information and Communication Technology, Springer Berlin Heidelberg, pp. 297–311, [online] Available from: http://link.springer.com/chapter/10.1007/978-3-642-15506-2_21 (Accessed 2 November 2015).

Guo, Yinghua and Slay, Jill (2010d) 'Testing Forensic Copy Function of Computer Forensics Investigation Tools', *Journal of Digital Forensic Practice*, 3(1), pp. 46–61.

Guttman, Barbara (2009) 'Digital Evidence Standards: Computer and Mobile Forensic Standards', In Albany, NY, USA, [online] Available from: http://d-forensics.org/2009/ (Accessed 1 April 2016).

Harley, D. (2012) 'AMTSO: The test of time?', *Network Security*, 2012(1), pp. 5–10.

Hex-Rays (2015) 'IDA: About', [online] Available from: https://www.hex-rays.com/products/ida/ (Accessed 11 May 2016).

HMSO (1990) 'Computer Misuse Act 1990', [online] Available from: http://www.legislation.gov.uk/ukpga/1990/18/contents (Accessed 25 May 2011).

HMSO (1984) 'Police and Criminal Evidence Act 1984', [online] Available from: http://www.legislation.gov.uk/ukpga/1984/60/section/69/enacted (Accessed 30 March 2016).

HMSO (1999) 'Youth Justice and Criminal Evidence Act 1999', [online] Available from: http://www.legislation.gov.uk/ukpga/1999/23/section/60 (Accessed 30 March 2016).

HMSO, Expert (1998) 'Data Protection Act 1998', [online] Available from: http://www.legislation.gov.uk/ukpga/1998/29/introduction (Accessed 10 May 2016).

Home Office (2013) 'Consultation on new statutory powers for the forensic science regulator', [online] Available from: https://www.gov.uk/government/uploads/system/uploads/attachment_data/file/25661

4/New_statutory_powers_for_the_forensic_science_regulator.pdf (Accessed 30 July 2014).

House of Commons (2016) 'Forensic Science Strategy:Standards and accreditation', *House of Commons - Forensic Science Strategy - Science and Technology Committee*, [online] Available from: https://www.publications.parliament.uk/pa/cm201617/cmselect/cmsctech/501/50107.htm (Accessed 1 February 2017).

House of Commons (2013) 'House of Commons - Forensic science - Science and Technology Committee', *Forensic science - Science and Technology Committee*, [online] Available from: http://www.publications.parliament.uk/pa/cm201314/cmselect/cmsctech/610/61006.htm (Accessed 30 March 2016).

Hubbard, Douglas W. (2014) *How to Measure Anything: Finding the Value of Intangibles in Business*, John Wiley & Sons.

Huber, Peter W. (1993) *Galileo's Revenge: Junk Science in the Courtroom*, Reprint 1993. Basic Books.

Huda, Shamsul, Abawajy, Jemal, Alazab, Mamoun, Abdollalihian, Mali, et al. (2016) 'Hybrids of support vector machine wrapper and filter based framework for malware detection', *Future Generation Computer Systems*, 55, pp. 376–390.

Hungenberg, Thomas and Eckert, Matthias (2016) 'INetSim: Internet Services Simulation Suite', *INetSim: Internet Services Simulation Suite*, [online] Available from: http://www.inetsim.org/ (Accessed 10 April 2016).

Hunton, Paul (2012) 'Managing the technical resource capability of cybercrime investigation: a UK law enforcement perspective', *Public Money & Management*, 32(3), pp. 225–232.

Ianelli, Nicholas, Kinder, Ross and Roylo, Christian (2007) *The Use of Malware Analysis in Support of Law Enforcement*, CERT Coordination Center, Carnegie Mellon University, [online] Available from: http://www.securitynewsportal.com/securitynews/article.php?title=The_Use_of_Malware_Ana lysis_in_Support_of_Law_Enforcement (Accessed 16 May 2010).

IBM (2016) 'IBM SPSS - IBM Analytics', [online] Available from: http://www.ibm.com/analytics/us/en/technology/spss/ (Accessed 13 February 2017).

IEEE (2005) 'IEEE Standard for Software Verification and Validation', *IEEE Std 1012-2004 (Revision of IEEE Std 1012-1998)*.

Intel (2016) 'Intel® 64 and IA-32 Architectures Software Developer's Manual', Intel Corporation, [online] Available from: http://www.intel.co.uk/content/dam/www/public/us/en/documents/manuals/64-ia-32-architectures-software-developer-instruction-set-reference-manual-325383.pdf.

ISO (2005) 'ISO/IEC 17025:2005 - General requirements for the competence of testing and calibration laboratories', *International Standards Organisation*, [online] Available from: http://www.iso.org/iso/catalogue_detail.htm?csnumber=39883 (Accessed 11 March 2011).

ISO (2015) 'ISO/IEC 27041:2015 - Information technology -- Security techniques -- Guidance on assuring suitability and adequacy of incident investigative method', International Organization for Standardization, [online] Available from: http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=44405 (Accessed 6 April 2016).

JCGM (2008) 'International Vocabulary of Metrology – Basic and General Concepts and Associated Terms VIM, 3rd edition, JCGM 200:2008', JCGM, [online] Available from: http://www.bipm.org/en/publications/guides/vim.html (Accessed 8 May 2011).

Joe Security (2017) 'Automated Malware Analysis - Joe Sandbox Cloud', [online] Available from: https://www.joesecurity.org/joe-sandbox-cloud (Accessed 15 February 2017).

Kaur, Ravneet and Kaur, Amandeep (2012) 'Digital forensics', *International Journal of Computer Applications*, 50(5).

Kennedy, Donald (2003) 'Forensic Science: Oxymoron?', *Science*, 5th December.

Kent, Karen, Chevalier, Suzanne, Grance, Timothy and Dang, Hung (2006) *SP 800-86. Guide to Integrating Forensic Techniques into Incident Response*, Gaithersburg, MD, United States, National Institute of Standards & Technology.

Kessler, G. C. (2007) 'Anti-forensics and the digital investigator', In *Proceedings of The 5 th Australian Digital Forensics Conference*, Citeseer, p. 1.

Kim, A.C., Kim, S., Park, W.H. and Lee, D.H. (2014) 'Fraud and financial crime detection model using malware forensics', *Multimedia Tools and Applications*, 68(2), pp. 479–496.

Kipling, Lesley (2012) 'The Sting in the Tail: The Trojan Horse Defence. A Proposed Methodology for detecting Indicators of Malware Compromise.', MSc, Cranfield University.

Kirat, Dhilung, Vigna, Giovanni and Kruegel, Christopher (2011) 'BareBox: Efficient Malware Analysis on Bare-metal', In *Proceedings of the 27th Annual Computer Security Applications Conference*, ACSAC '11, New York, NY, USA, ACM, pp. 403–412, [online] Available from: http://doi.acm.org/10.1145/2076732.2076790 (Accessed 22 March 2016).

Košinár, Peter, Malcho, Juraj, Marko, Richard and Harley, David (2010) 'AV testing exposed', In Vancouver.

Krister, K. M. (2009) 'Automated Analyses of Malicious Code', MSc, Oslo, Norwegian University of Science and Technology.

Kritzer, H. M. (2009) 'The arts of persuasion in science and law: Conflicting norms in the courtroom', *Law and contemporary problems*, 72(1), pp. 41–61.

Kubi, A. K., Saleem, S. and Popov, O. (2011) 'Evaluation of some tools for extracting e-evidence from mobile devices', In *2011 5th International Conference on Application of Information and Communication Technologies (AICT)*, pp. 1–6.

Law Commission (2011) *Expert Evidence in Criminal Proceedings in England and Wales*,.

Lempereur, Brett, Merabti, Madjid and Shi, Qi (2010) 'Pypette: A framework for the automated evaluation of live digital forensic techniques', In *Proceedings of the 11th Annual PostGraduate Symposium on The Convergence of Telecommunications Networking and Broadcasting*.

Liang, James (2010) 'Evaluating a selection of tools for extraction of forensic data: disk imaging', Thesis, Auckland University of Technology, [online] Available from: http://aut.researchgateway.ac.nz/handle/10292/1204 (Accessed 8 February 2016).

Liao, Yi-Ching and Langweg, Hanno (2014) 'A survey of process activity tracking system', *Norsk informasjonssikkerhetskonferanse (NISK)*, 2013.

Ligh, Michael, Adair, Steven, Hartstein, Blake and Richard, Matthew (2010) *Malware Analyst's Cookbook and DVD: Tools and Techniques for Fighting Malicious Code*, Pap/Dvdr edition. Indianapolis, IN, John Wiley & Sons.

Limongelli, V. (2008) 'Digital Evidence: Findings of Reliability, Not Presumptions', *Journal of Digital Forensic Practice*, 2(1), pp. 13–16.

Lloyd, Ian J. (2014) *Information Technology Law*, Oxford University Press.

Lyda, Robert and Hamrock, James (2006) 'Exploring investigative methods for identifying and profiling serial bots', *Journal of Digital Forensic Practice*, 1(3), pp. 165–177.

Lyle, James R. (2010) 'If error rate is such a simple concept, why don't I have one for my forensic tool yet?', *Digital Investigation*, 7(Supplement 1), pp. S135–S139.

Malin, C. H., Casey, E. and Aquilina, J. M. (2008) *Malware forensics: investigating and analyzing malicious code*, Syngress Publishing.

Malin, Cameron H., Casey, Eoghan and Aquilina, James M. (2013) *Malware Forensics Field Guide for Linux Systems: Digital Forensics Field Guides*, Elsevier.

Malin, Cameron H., Casey, Eoghan and Aquilina, James M. (2012) *Malware Forensics Field Guide for Windows Systems: Digital Forensics Field Guides*, Elsevier.

Malwr (2016) 'Malwr - Malware Analysis by Cuckoo Sandbox', [online] Available from: https://malwr.com/ (Accessed 23 August 2016).

Marshall, Angus M. (2010) 'Quality Standards and Regulation: Challenges for Digital Forensics', *Measurement and Control*, 43(8), pp. 243–247.

Marshall, Angus M. (2011) 'Standards, regulation & quality in digital investigations: The state we are in', *digital investigation*, 8(2), pp. 141–144.

Marsico, Christopher V (2004) *Computer Evidence v. Daubert: The coming conflict*, CERIAS, Tech report, West Lafayette, IN, USA, Purdue University School of Technology, [online] Available from: https://www.cerias.purdue.edu/assets/pdf/bibtex_archive/2005-17.pdf (Accessed 17 February 2012).

Martignoni, L., Paleari, R. and Bruschi, D. (2009) 'A Framework for Behavior-Based Malware Analysis in the Cloud', In *Information Systems Security: 5th International Conference, ICISS 2009 Kolkata, India, December 14-18, 2009 Proceedings*, Springer, p. 178.

McLinden, Sean (2009) ‘Child Porn Virus’, *Guidance Software Inc.*, Closed forum, [online] Available from: https://support.guidancesoftware.com/forum/showthread.php?t=36363&highlight=child+porn+virus (Accessed 5 April 2011).

McLinden, Sean (2011) ‘Email to Ian Kennedy, 22 January’,.

Merriam-Webster (2017) ‘evaluate, v.’, Merriam-Webster, [online] Available from: https://www.merriam-webster.com/dictionary/evaluate.

Ministry of Justice (1999) ‘Civil Procedure Rules 1998’, [online] Available from: http://www.legislation.gov.uk/uksi/1998/3132/contents/made (Accessed 21 February 2017).

Ministry of Justice (2015) ‘Criminal Procedure Rules 2015’, procedure rules, [online] Available from: http://www.justice.gov.uk/guidance/courts-and-tribunals/courts/procedure-rules/criminal/rulesmenu.htm (Accessed 21 February 2017).

Mohurle, Shubham, Khutwad, Seema, Kunjir, Pratiksha and Bhosle, Anjali (2016) ‘Review Paper on Ear Biometric Authentication’, *International Journal of Engineering Science*, 2875, [online] Available from: http://ijesc.org/upload/8d0addaf9092120e3dbca77217eda5a6.Review%20Paper%20on%20Ear%20Biometric%20Authentication.pdf (Accessed 21 February 2017).

Montasari, Reza, Peltola, Pekka and Evans, David (2015) ‘Integrated Computer Forensics Investigation Process Model (ICFIPM) for Computer Crime Investigations’, In Jahankhani, H., Carlile, A., Akhgar, B., Taal, A., et al. (eds.), *Global Security, Safety and Sustainability: Tomorrow’s Challenges of Cybersecurity*, Communications in Computer and Information Science, Springer International Publishing, pp. 83–95, [online] Available from: http://link.springer.com/chapter/10.1007/978-3-319-23276-8_8 (Accessed 7 March 2016).

Morales, J.A., Sandhu, R. and Xu, S. (2010) ‘Evaluating detection and treatment effectiveness of commercial anti-malware programs’, In pp. 31–38.

Moser, A., Kruegel, C. and Kirda, E. (2007) ‘Exploring Multiple Execution Paths for Malware Analysis’, In *IEEE Symposium on Security and Privacy, 2007. SP ’07*, pp. 231–245.

Mundie, D.A. and Mcintire, D.M. (2013) ‘An Ontology for Malware Analysis’, In *2013 Eighth International Conference on Availability, Reliability and Security (ARES)*, pp. 556–558.

Namanya, Anitta Patience, Pagna-Disso, Jules and Awan, Irfan (2015) ‘Evaluation of automated static analysis tools for malware detection in Portable Executable files’, In *31st UK Performance Engineering Workshop 17 September 2015*, p. 81.

National Audit Office (2013) ‘The UK cybersecurity strategy: Landscape review - National Audit Office (NAO)’, *National Audit Office*, [online] Available from: https://www.nao.org.uk/report/the-uk-cyber-security-strategy-landscape-review/ (Accessed 3 November 2015).

Newsham, Tim, Palmer, Chris, Stamos, Alex and Burns, Jesse (2007) ‘Breaking Forensics Software - Flaws in Critical Evidence Collection’, ISEC Partners, [online] Available

from: http://www.isecpartners.com/storage/white-papers/iSEC-Breaking_Forensics_Software-Paper.v1_1.BH2007.pdf (Accessed 18 October 2011).

NIST (2003a) 'CFTT Methodology Overview', [online] Available from: http://www.cftt.nist.gov/Methodology_Overview.htm (Accessed 4 April 2016).

NIST (2005) 'Digital Data Acquisition Tool Test Assertions and Test Plan', National Institute of Standards and Technology, [online] Available from: http://www.cftt.nist.gov/DA-ATP-pc-01.pdf.

NIST (2003b) 'NIST Computer Forensic Tool Testing Program', *NIST Computer Forensic Tool Testing Program*, [online] Available from: http://www.cftt.nist.gov/ (Accessed 27 May 2011).

NIST (2002) *Test Results for Disk Imaging Tools: dd GNU fileutils 4.0.36, Provided with Red Hat Linux 7.1*, National Institute of Standards & Technology, [online] Available from: https://www.justnet.org/pdf/196352.pdf (Accessed 7 April 2016).

NIST (2016) 'The CFReDS Project', *The CFReDS Project*, [online] Available from: http://www.cfreds.nist.gov/ (Accessed 23 March 2016).

OED (2016a) 'evaluate, v.', *OED Online*, Oxford University Press, [online] Available from: http://www.oed.com/view/Entry/65181 (Accessed 29 March 2016).

OED (2016b) 'risk, n.', *OED Online*, Oxford University Press, [online] Available from: http://www.oed.com.libezproxy.open.ac.uk/view/Entry/166306 (Accessed 7 April 2016).

OLAF (2016) 'Guidelines on Digital Forensic Procedures for OLAF Staff', European Anti-fraud Office (OLAF), [online] Available from: https://ec.europa.eu/anti-fraud/sites/antifraud/files/guidelines_en.pdf (Accessed 7 February 2017).

Palkmets, Lauri, Ciobanu, Cosmin, Leguesse, Yonas and Sidiropoulos, Christos (2014) 'Building artifact handling and analysis environment toolset', European Union Agency for Network and Information Security (ENISA), [online] Available from: https://www.enisa.europa.eu/topics/trainings-for-cybersecurity-specialists/online-training-material/documents/building-artifact-handling-and-analysis-environment-toolset/view (Accessed 9 May 2016).

Palmer, Gary (2001) 'A road map for digital forensic research', In *First Digital Forensic Research Workshop (DFRWS)*, Utica, New York, pp. 27–30.

Pan, Lei and Batten, Lynn M. (2009) 'Robust performance testing for digital forensic tools', *Digital Investigation*, 6(1–2), pp. 71–81.

Pan, Y., Schwartz, D. and Mishra, S. (2015) 'Gamified digital forensics course modules for undergraduates', In *2015 IEEE Integrated STEM Education Conference (ISEC)*, pp. 100–105.

Panik, Michael J. (2005) *Advanced Statistics from an Elementary Point of View*, Academic Press.

Payload Security (n.d.) 'Free Automated Malware Analysis Service - powered by VxStream Sandbox', *Payload Security*, [online] Available from: https://www.hybrid-analysis.com/ (Accessed 15 August 2016).

Pearce, Michael, Zeadally, Sherali and Hunt, Ray (2013) 'Virtualization: Issues, Security Threats, and Solutions', *ACM Comput. Surv.*, 45(2), p. 17:1–17:39.

Peisert, S. and Bishop, M. (2007) 'How to Design Computer Security Experiments', In *Fifth World Conference on Information Security Education*, Springer, pp. 141–148.

Peisert, Sean, Bishop, Matt and Marzullo, Keith (2008) 'Computer forensics *in forensis*', *SIGOPS Oper. Syst. Rev.*, 42(3), pp. 112–122.

Pollitt, M. M. (2007) 'An Ad Hoc Review of Digital Forensic Models', In *Second International Workshop on Systematic Approaches to Digital Forensic Engineering, 2007. SADFE 2007*, pp. 43–54.

Pollitt, Mark (2010) 'A History of Digital Forensics', In Chow, K.-P. and Shenoi, S. (eds.), *Advances in Digital Forensics VI - Sixth IFIP WG 11.9 International Conference on Digital Forensics*, IFIP Advances in Information and Communication Technology, Hong Kong, China, Springer Berlin Heidelberg, pp. 3–15, [online] Available from: http://link.springer.com/10.1007/978-3-642-15506-2 (Accessed 8 March 2016).

Popper, Karl R (1968) *The Logic of Scientific Discovery*, Rev ed. London, Hutchinson.

Potter, Bruce and Day, Greg (2009) 'The effectiveness of anti-malware tools', *Computer Fraud & Security*, 2009(3), pp. 12–13.

Provataki, Athina and Katos, Vasilios (2013) 'Differential malware forensics', *Digital Investigation*, 10(4), pp. 311–322.

Ragan, Steve (2008) 'Malware caused kiddie porn investigation reveals', *The Tech Herald*, News, [online] Available from: http://www.thetechherald.com/articles/Malware-caused-kiddie-porn-investigation-reveals/612/ (Accessed 30 November 2015).

Raghavan, Sriram (2012) 'Digital forensic research: current state of the art', *CSI Transactions on ICT*, 1(1), pp. 91–114.

Raphel, Jithu and Vinod, P. (2015) 'Information Theoretic Method for Classification of Packed and Encoded Files', In *Proceedings of the 8th International Conference on Security of Information and Networks*, SIN '15, New York, NY, USA, ACM, pp. 296–303, [online] Available from: http://doi.acm.org/10.1145/2799979.2800015 (Accessed 14 March 2016).

Rasch, Mark (2007) 'Was Julie Amero wrongly convicted?', *The Register*, [online] Available from: http://www.theregister.co.uk/2007/02/14/julie_amero_case/page2.html (Accessed 22 November 2011).

Rieck (2008) 'Learning and classification of malware behavior', Springer.

Robertson, Suzanne and Robertson, James (2012) *Mastering the Requirements Process: Getting Requirements Right*, Addison-Wesley.

Ross, J. (2010) 'Malware Analysis for the Enterprise', In *Black Hat DC 2010*, [online] Available from: http://www.blackhat.com/presentations/bh-dc-10/Powell_Shane/BlackHat-DC-2010-Powell-Cyber-Effects-Prediction-wp.pdf.

Royal, P., Halpin, M., Dagon, D., Edmonds, R. and Lee, W. (2006) 'PolyUnpack: Automating the Hidden-Code Extraction of Unpack-Executing Malware', In

Computer Security Applications Conference, 2006. ACSAC '06. 22nd Annual, pp. 289–300.

Royal Statistical Society (2001) 'The Royal Statistical Society', [online] Available from: http://www.rss.org.uk/site/cms/contentviewarticle.asp?article=527 (Accessed 13 November 2011).

Royce, Winston W. (1970) 'Managing the development of large software systems', In *proceedings of IEEE WESCON*, Los Angeles, pp. 328–338, [online] Available from: http://sites.google.com/site/yamilejaime/ArtclasRoyce1.pdf (Accessed 9 October 2016).

Runciman, Brian (2011) 'Malware Response', *ITNOW*, 53(6), pp. 34–36.

Russinovich, Mark (2016) 'Process Monitor', *Process Monitor*, [online] Available from: https://technet.microsoft.com/en-us/sysinternals/processmonitor.aspx (Accessed 24 October 2016).

Russinovich, Mark (2011) 'TCPView for Windows', *TCPView*, [online] Available from: https://technet.microsoft.com/en-us/sysinternals/tcpview.aspx (Accessed 24 October 2016).

Saks, Michael J. and Faigman, David L. (2008) 'Failed Forensics: How Forensic Science Lost Its Way and How It Might Yet Find It', *Annual Review of Law and Social Science*, 4(1), pp. 149–171.

Saleem, Shahzad, Popov, Oliver and Appiah-Kubi, Oheneba Kwame (2012) 'Evaluating and Comparing Tools for Mobile Device Forensics Using Quantitative Analysis', In Rogers, M. and Seigfried-Spellar, K. C. (eds.), *Digital Forensics and Cyber Crime*, Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, Springer Berlin Heidelberg, pp. 264–282, [online] Available from: http://link.springer.com/chapter/10.1007/978-3-642-39891-9_17 (Accessed 18 January 2016).

Sanderson, Paul (2008) 'Encase wraps files > 4GB in size when viewed in Mounted Network Shares', SandersonForensics, [online] Available from: http://www.sandersonforensics.com/Files/Encase%20mounted%20wrap%20bug.pdf (Accessed 5 March 2016).

Seifert, Christian, Steenson, Ramon, Welch, Ian, Komisarczuk, Peter and Endicott-Popovsky, Barbara (2007) 'Capture – A behavioral analysis tool for applications and documents', *Digital Investigation*, 4, Supplement, pp. 23–30.

Shanmugam, Karthikeyan (2011) 'Validating digital forensic evidence', Thesis, Brunel University School of Engineering and Design PhD Theses, [online] Available from: http://bura.brunel.ac.uk/handle/2438/7651 (Accessed 2 November 2015).

Sheskin, David J. (2011) *Handbook of Parametric and Nonparametric Statistical Procedures, Fifth Edition*, 5 edition. Boca Raton, Chapman and Hall/CRC.

Shijo, P. V. and Salim, A. (2015) 'Integrated Static and Dynamic Analysis for Malware Detection', *Procedia Computer Science*, Proceedings of the International Conference on Information and Communication Technologies, ICICT 2014, 3-5 December 2014 at Bolgatty Palace & Island Resort, Kochi, India, 46, pp. 804–811.

Shosha, A. F., Tobin, L. and Gladyshev, P. (2013) 'Digital Forensic Reconstruction of a Program Action', In *2013 IEEE Security and Privacy Workshops (SPW)*, pp. 119–122.

Sikorski, Michael and Honig, Andrew (2012) *Practical Malware Analysis: The Hands-On Guide to Dissecting Malicious Software*, 1 edition. San Francisco, No Starch Press.

Smith, Martha (2012) 'Factors Influencing Power', *Common Mistakes in using statistics*, [online] Available from: https://www.ma.utexas.edu/users/mks/statmistakes/FactorsInfluencingPower.html (Accessed 23 August 2016).

Smith, Reginald D. (2014) 'Malware "Ecology" Viewed as Ecological Succession: Historical Trends and Future Prospects', *arXiv:1410.8082 [cs, q-bio]*, [online] Available from: http://arxiv.org/abs/1410.8082 (Accessed 14 November 2015).

Solicitors Journal (2011) 'Solicitors Journal - New reliability test for experts "would prevent miscarriages of justice"', *Solictors Journal*, [online] Available from: http://www.solicitorsjournal.com/story.asp?sectioncode=2&storycode=18001&c=3&eclipse_action=getsession (Accessed 4 May 2011).

Sommer, P (2011) 'Certification, registration and assessment of digital forensic experts: The UK experience', *DIGITAL INVESTIGATION*, 8(2), pp. 98–105.

Sommer, Peter (2010) 'Forensic science standards in fast-changing environments☆', *Science & Justice*, 50(1), pp. 12–17.

Stevens, S. S. (1946) 'On the Theory of Scales of Measurement', *Science*, 103(2684), pp. 677–680.

Studd, Helen (2003) 'Mother "killed three of her babies, then blamed cot death"', *The Times*, London, England, 30th April.

Sukwong, O., Kim, H.S. and Hoe, J.C. (2011) 'Commercial Antivirus Software Effectiveness: An Empirical Study', *Computer*, 44(3), pp. 63–70.

Supreme Court (2011) *Jones (Appellant) v Kaney (Respondent) [2011] UKSC 13*,.

Sutherland, I., Evans, J., Tryfonas, T. and Blyth, A. (2008) 'Acquiring volatile operating system data tools and techniques', *ACM SIGOPS Operating Systems Review*, 42(3), pp. 65–73.

SWGDE (2012) 'SWGDE Model Standard Operation Procedures for Computer Forensics', Scientific Working Group on Digital Evidence (SWGDE), [online] Available from: https://www.swgde.org/documents/Current+Documents/SWGDE+QAM+and+SOP+Manuals/2012-09-13+SWGDE+Model+SOP+for+Computer+Forensics+v3.

SWGDE (2008) 'SWGDE Standards and Controls Position Paper, v1.0', Scientific Working Group on Digital Evidence (SWGDE), [online] Available from: https://www.swgde.org/documents/Current%20Documents/2008-01-30%20SWGDE%20Position%20Paper%20Standards%20and%20Controls%20v1.0 (Accessed 4 February 2016).

Szor, Peter (2005) *The Art of Computer Virus Research and Defense*, 01 edition. Upper Saddle River, NJ, Addison-Wesley Professional.

The Times (2003) ‘Teenager cleared of US internet attack’, *The Times*, London, England, 18th October.

ThreatExpert (2011) ‘ThreatExpert - Submit Your Sample Online’, *ThreatExpert - Automated Threat Analysis*, [online] Available from: http://www.threatexpert.com/submit.aspx (Accessed 8 April 2011).

ThreatTrack Security (2016) ‘Malware Analysis Tool, Dynamic Malware Sandbox - ThreatAnalyzer - ThreatTrack’, [online] Available from: https://www.threattrack.com/malware-analysis.aspx (Accessed 23 August 2016).

Tichy, Walter F. (1998) ‘Should computer scientists experiment more?’, *IEEE Computer*, (5), pp. 32–40.

Toner, John (2017) ‘Forensic science standards under threat says regulator’, *Police Oracle*, UK Police News, [online] Available from: http://www.policeoracle.com/news/Forensic-science-standards-under-threat-says-regulator-_93811.html (Accessed 2 February 2017).

Traore, Issa, Awad, Ahmed and Woungang, Isaac (2017) *Information Security Practices: Emerging Threats and Perspectives*, Springer.

Turner, Philip (2008) ‘Digital Evidence Bags’, PhD, Oxford Brookes University.

Underwriters Laboratory (2016) ‘UL : Software and Security’, *Software and security*, [online] Available from: industries.ul.com/software-and-security (Accessed 9 April 2016).

Van Buskirk, Eric and Liu, Vincent T. (2006) ‘Digital Evidence: Challenging the Presumption of Reliability’, *Journal of Digital Forensic Practice*, 1(1), pp. 19–26.

Vincze, Eva A. (2016) ‘Challenges in digital forensics’, *Police Practice and Research*, 17(2), pp. 183–194.

VirusTotal (2010) ‘VirusTotal - Free Online Virus and Malware Scan’, *VirusTotal - Free Online Virus, Malware and URL Scanner*, [online] Available from: http://www.virustotal.com/ (Accessed 14 November 2011).

VMWare (2016) ‘VMware Virtualization for Desktop & Server, Application, Public & Hybrid Clouds | VMware United Kingdom’, [online] Available from: http://www.vmware.com/uk (Accessed 10 May 2016).

Wagener, G., Dulaunoy, A. and Engel, T. (2008) ‘An Instrumented Analysis of Unknown Software and Malware Driven by Free Libre Open Source Software’, In *IEEE International Conference on Signal Image Technology and Internet Based Systems, 2008. SITIS ’08*, pp. 597–605.

Wang, Tzy-Shiah, Lin, Hui-Tang, Cheng, Wei-Tsung and Chen, Chang-Yu (2017) ‘DBod: Clustering and detecting DGA-based botnets using DNS traffic analysis’, *Computers & Security*, 64, pp. 1–15.

Welham, Jamie (2010) ‘Expert reveals “flaws” in child porn inquiry’, *Camdem New Journal*, Newspaper, [online] Available from: http://www.camdennewjournal.com/news/2010/oct/expert-reveals-%E2%80%98flaws%E2%80%99-child-porn-inquiry (Accessed 30 November 2015).

Willems, Carsten, Holz, Thorsten and Freiling, Felix (2007) 'Toward Automated Dynamic Malware Analysis Using CWSandbox', *IEEE Security and Privacy Magazine*, 5(2), p. 32 39.

Williams, Janet (2012) 'ACPO Good Practice Guide for Digital Evidence v5.0', ACPO.

Wilsdon, T. and Slay, J. (2005) 'Digital forensics: exploring validation, verification & certification', In *Systematic Approaches to Digital Forensic Engineering, 2005. First International Workshop on*, pp. 48–55.

Wilsdon, Tom and Slay, Jill (2006) 'Validation of Forensic Computing Software Utilizing Black Box Testing Techniques', In *Proceedings of the 4th Australian Digital Forensics Conference*, Edith Cowan University, Perth, Western Australia, School of Computer and Information Science Edith Cowan University Perth, Western Australia, [online] Available from: http://ro.ecu.edu.au/adf/37 (Accessed 10 April 2011).

Wilson, Craig (2011) 'Digital Evidence Discrepancies – Casey Anthony Trial « Digital Detective Blog – Digital Forensic Analysis and Data Recovery', *Digital-Detective*, [online] Available from: http://wordpress.bladeforensics.com/?p=357 (Accessed 8 December 2011).

Wueest, Candid (2015) 'Does malware still detect virtual machines?', *Symantec Security Response*, [online] Available from: http://www.symantec.com/connect/blogs/does-malware-still-detect-virtual-machines (Accessed 23 February 2016).

Zareen, M. S., Waqar, A. and Aslam, B. (2013) 'Digital forensics: Latest challenges and response', In *2013 2nd National Conference on Information Assurance (NCIA)*, pp. 21–29.

Zolkipli, Mohamad Fadli and Jantan, Aman (2011) 'A framework for defining malware behavior using run time analysis and resource monitoring', In *Communications in Computer and Information Science*, Heidelberg, Springer, pp. 199–209.

APPENDIX A Literature review sources

The following sources were used to perform the literature review for this research.

Source type	**Source**
Databases	IEEE
	Science Direct
	Lexis-Nexis
	Springer-link
	Taylor and Francis
	Scopus
Social media	http://www.icerocket.com/
	http://www.h-net.org/
Mail lists	HTCC
	http://lsoft.com/lists/list_q.html
	http://www.jiscmail.ac.uk/
Discussion groups/Usenet	https://groups.google.com/forum/#!browse
	http://nzbindex.com/
Official reports/transcripts	http://researchbriefings.parliament.uk/
	www.official-documents.co.uk
	http://europa.eu/
	www.statistics.gov.uk
	http://ec.europa.eu/eurostat
Datasets	https://data.gov.uk/
Dissertations	Proquest (via OU library)
	http://oaister.worldcat.org/ - filter on Thesis
Search engines	http://oaister.worldcat.org/
	Google scholar
Forums	http://boardreader.com/

APPENDIX B List of Test runs performed

Test	Data folder	Test START	Test END	Test LENGTH[1]	Analysis START	Analysis END	Analysis	Duration	Tool	Pass	Average	Average
149	061016.2221	06/10/16 22:21	07/10/16 11:17	12:56	07/10/16 17:40	07/10/16 23:06	5:26	10 sec	Process Monitor	3	13:31	05:28
148	051016.1811	05/10/16 18:11	06/10/16 08:29	14:18	07/10/16 17:39	07/10/16 23:01	5:22	10 sec	Process Monitor	2		
147	041016.2020	04/10/16 20:20	05/10/16 09:39	13:19	07/10/16 17:39	07/10/16 23:14	5:35	10 sec	Process Monitor	1		
152	111016.0704	11/10/16 07:04	11/10/16 20:33	13:29	11/10/16 22:17	12/10/16 03:08	4:51	10 sec	TCPVCon	3	13:11	04:49
151	091016.1933	09/10/16 19:33	10/10/16 09:19	13:46	11/10/16 22:16	12/10/16 02:57	4:40	10 sec	TCPVCon	2		
150	081016.1700	08/10/16 17:00	09/10/16 05:17	12:17	11/10/16 22:16	12/10/16 03:13	4:56	10 sec	TCPVCon	1		
127	170816.1011	17/08/16 10:11	18/08/16 01:44	15:33	18/08/16 10:07	18/08/16 16:44	6:37	01 min	Process Monitor	3	15:26	06:19
126	160816.1749	16/08/16 17:49	17/08/16 08:30	14:41	17/08/16 10:20	17/08/16 17:41	7:20	01 min	Process Monitor	2		
114	180716.1712	18/07/16 17:12	19/07/16 09:16	16:04	20/07/16 09:21	20/07/16 14:20	4:59	01 min	Process Monitor	1		
129	190816.0048	19/08/16 00:48	19/08/16 12:56	12:08	20/08/16 08:47	20/08/16 13:34	4:46	01 min	TCPVCon	3	13:09	04:39
128	180816.0955	18/08/16 09:55	18/08/16 23:37	13:42	20/08/16 08:47	20/08/16 13:46	4:59	01 min	TCPVCon	2		
115	190716.1356	19/07/16 13:56	20/07/16 03:33	13:37	22/07/16 12:12	22/07/16 16:24	4:12	01 min	TCPVCon	1		
135	270816.1715	27/08/16 17:15	28/08/16 13:58	20:43	29/08/16 11:12	29/08/16 18:28	7:16	05 min	Process Monitor	3	21:44	06:00
134	250816.1334	25/08/16 13:34	26/08/16 13:54	24:20	29/08/16 11:11	29/08/16 18:27	7:15	05 min	Process Monitor	2		
117	250716.1048	25/07/16 10:48	26/07/16 06:56	20:08	26/07/16 14:48	26/07/16 18:17	3:29	05 min	Process Monitor	1		
131	210816.1006	21/08/16 10:06	22/08/16 07:52	21:46	22/08/16 20:48	23/08/16 02:07	5:19	05 min	TCPVCon	3	20:37	04:53
130	200816.0858	20/08/16 08:58	21/08/16 07:01	22:03	22/08/16 20:48	23/08/16 02:16	5:28	05 min	TCPVCon	2		
116	230716.1127	23/07/16 11:27	24/07/16 05:30	18:03	26/07/16 09:08	26/07/16 13:01	3:53	05 min	TCPVCon	1		
137	020916.2120	02/09/16 21:20	03/09/16 22:04	24:44	04/09/16 10:11	04/09/16 17:15	7:04	10 min	Process Monitor	3	22:48	06:13
136	010916.0814	01/09/16 08:14	02/09/16 10:58	26:44	04/09/16 10:11	04/09/16 17:09	6:58	10 min	Process Monitor	2		
118	270716.1542	27/07/16 15:42	28/07/16 08:38	16:56	31/07/16 20:35	01/08/16 01:12	4:36	10 min	Process Monitor	1		
133	230816.2250	23/08/16 22:50	25/08/16 00:00	25:10	25/08/16 10:50	25/08/16 16:29	5:38	10 min	TCPVCon	3	24:17	06:03
132	220816.2033	22/08/16 20:33	23/08/16 22:03	25:30	25/08/16 10:49	25/08/16 17:16	6:27	10 min	TCPVCon	2		
119	280716.1559	28/07/16 15:59	29/07/16 14:10	22:11	31/07/16 11:03	31/07/16 17:08	6:04	10 min	TCPVCon	1		

Note: 1 – Elapsed time reported as HH:MM